TUGAS AKHIR – KI141502

PENGEMBANGAN SISTEM MONITORING AKTIVITAS FISIK USER BERGERAK DENGAN ANALISA LANGKAH (*STEP ANALYSIS*) UNTUK ESTIMASI PEMBAKARAN KALORI SECARA *REAL-TIME* MENGGUNAKAN *MICROCONTROLLER* ARDUINO

ACHMAD FAUZI AZMI
NRP 5112100192

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – KI141502

DEVELOPMENT OF MONITORING SYSTEM FOR PHYSICAL ACTIVITY OF MOBILE USERS BASED ON STEP ANALYSIS TO ESTIMATE REAL-TIME BURNING CALORIES USING ARDUINO MICROCONTROLLER

ACHMAD FAUZI AZMI
NRP 5112100192

Supervisor I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Supervisor II
Ir. Muchammad Husni, M.Kom

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENGEMBANGAN SISTEM MONITORING AKTIVITAS FISIK USER BERGERAK DENGAN ANALISA LANGKAH (*STEP ANALYSIS*) UNTUK ESTIMASI PEMBAKARAN KALORI SECARA *REAL-TIME* MENGGUNAKAN *MICROCONTROLLER* ARDUINO

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
ACHMAD FAUZI AZMI
NRP : 5112 100 192

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Waskitho W., S.Kom., M.Eng., Ph.D. .....
   NIP: 19741022 200003 1 001 (Pembimbing 1)

2. Ir. Muchammad Husni, M.Kom. .....
   NIP: 19600221198403 1 001 (Pembimbing 2)

SURABAYA
JUNI, 2016

[Halaman ini sengaja dikosongkan]

PENGEMBANGAN SISTEM MONITORING AKTIVITAS FISIK USER BERGERAK DENGAN ANALISA LANGKAH (*STEP ANALYSIS*) UNTUK ESTIMASI PEMBAKARAN KALORI SECARA *REAL-TIME* MENGGUNAKAN *MICROCONTROLLER* ARDUINO

Nama Mahasiswa	**: Achmad Fauzi Azmi**
NRP	**: 5112100192**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Ir. Muchammad Husni, M.Kom.**

Abstrak

Demi menjaga agar tubuh tetap sehat adalah dengan cara mengatur aktivitas sehari-hari seperti pola istirahat, olahraga, makan dan sebagainya. Namun tidak dipungkiri dalam memonitor semua aktivitas tersebut kadang anda dihambat oleh rasa malas atau lupa. Dibutuhkan suatu sistem untuk memantau aktivitas fisik secara wearable.

Perangkat wearable ini memanfaatkan mikrokontroler arduino, modul bluetooth, li-po battery dan sensor akselerometer ADXL 345 untuk membuat sebuah jaringan sensor nirkabel sederhana, yang dapat digunakan untuk mengkalkulasikan data jumlah langkah, jarak yang ditempuh dan pembakaran kalori.

Dari beberapa uji coba perangkat wearable tersebut, sistem monitoring kalori dipakai pada kondisi berjalan di medan datar. Didapatkan tingkat akurasi jumlah langkah sebesar 92,22%. Untuk jarak, rata-rata akurasi dari jarak yang ditempuh didapatkan sebesar 86,64%. Dan untuk kalori, rata-rata presisi kalori yang terbakar dari hasil perangkat wearable

terhadap rata-rata hasil beberapa aplikasi android didapatkan sebesar 86,58%.

Kata kunci: Monitoring kalori, deteksi langkah, arduino, sensor akselerometer.

DEVELOPMENT OF MONITORING SYSTEM FOR PHYSICAL ACTIVITY OF MOBILE USERS BASED ON STEP ANALYSIS TO ESTIMATE REAL-TIME BURNING CALORIES USING ARDUINO MICROCONTROLLER

Nama Mahasiswa	**: Achmad Fauzi Azmi**
NRP	**: 5112100192**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S.Kom., M.Eng., PhD.**
Dosen Pembimbing 2	**: Ir. Muchammad Husni, M.Kom.**

Abstract

To keep body stay fit, you should maintain your daily activity, like taking a rest, workout, eating, etc. But, it's pretty hard to monitor your daily activity because sometimes you feel tired or lazy to do it. So, you need a wearable device which can track your daily activity.

This wearable device uses arduino microcontroller, accelerometer sensor (ADXL 345), bluetooth module and li-po battery to create a simple wireless sensor network that can be used to count steps, measured distance and calories burned.

From few experiments of this wearable device, the calories monitoring system used in walking conditions on flat terrain. Obtained the accuracy rate of steps is 92.22%. For the distance, the accuracy rate of measured distance obtained by 86.64%. And for the calories burned, the precision rate of calories burned from the results of wearable device compares to the rate results of few android aplications obtained by 86.58%.

Keywords: Calorie counter, Step detection, Arduino, Accelerometer Sensor.

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"PENGEMBANGAN SISTEM MONITORING AKTIVITAS FISIK USER BERGERAK DENGAN ANALISA LANGKAH (*STEP ANALYSIS*) UNTUK ESTIMASI PEMBAKARAN KALORI SECARA *REAL-TIME* MENGGUNAKAN *MICROCONTROLLER* ARDUINO"

Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Keluarga penulis yang telah memberikan dukungan moral dan material serta do'a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Bapak Ir. Muchammad Husni, M.Kom**.** selaku pembimbing II yang juga telah membantu, membimbing, dan

memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.

5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS.
6. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
7. Teman-teman angkatan 2012 yang yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis, kakak-kakak angkatan 2011 dan 2010 serta adik-adik angkatan 2013 dan 2014 yang membuat penulis untuk selalu belajar.
8. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, 23 Juni 2016
Penulis

Achmad Fauzi Azmi

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Tubuh yang sehat berpengaruh pada kinerja sehari-hari. Namun, untuk menjaga agar tubuh tetap sehat bukanlah perkara mudah. Pasalnya, kegiatan yang padat terkadang menyita waktu seseorang untuk melakukan olahraga. Oleh karena itu, mereka dapat menggunakan sebuah *gadget* untuk memantau kesehatan mereka dengan mengetahui seberapa besar kalori yang terbakar. Untuk menghitung kalori yang terbakar atau energi yang dikeluarkan tubuh manusia perlu kegiatan fisik. Aktivitas fisik dapat berupa gerakan tubuh manusia dengan otot yang membutuhkan energi.

Dalam penggunaan *gadget* menggunakan *3-axis accelerometer* ADXL 345 dengan fitur yang dapat menghitung dan mengenali langkah (*step detection*), mengukur jarak yang ditempuh, dan kalori yang terbakar. Faktor tinggi badan dan berat badan pengguna juga diperlukan untuk menetukan hasil dari fitur tersebut. Untuk menampilkan data dari fitur tersebut akan ditampilkan di *mobile*.

Penggunaan *gadget* itu memanfaatkan sensor *accelerometer* dengan mikrokontroler arduino dan perangkat *mobile* yang digunakan adalah *mobile* berbasis android, sedangkan untuk transimisi data akan menggunakan *bluetooth*.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana cara kerja dari sistem monitoring aktivitas fisik user bergerak dengan analisa langkah (Step Analysis) secara *real-time* menggunakan mikrokontroler Arduino yang akan dibuat?

2. Bagaimana rancangan bentuk rangkaian monitoring sistem monitoring aktivitas fisik user bergerak dengan analisa langkah (Step Analysis) secara real-time menggunakan mikrokontroler Arduino yang akan dibuat?
3. Apa saja komponen yang dibutuhkan untuk membuat rangkaian ini?
4. Bagaimana cara menampilkan data aktivitas fisik user bergerak pada perangkat bergerak (mobile)?
5. Bagaimana mengimplementasikan algoritma untuk menghitung dan mengenali langkah (Step Detection), mengukur jarak yang ditempuh, dan kalori yang terbakar?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

1. Teknologi perangkat bergerak (*mobile*) yang digunakan berbasis Android.
2. Jumlah langkah kaki, jarak yang ditempuh dan energi yang dikeluarkan tubuh dalam bentuk kalori, ditampilkan di perangkat bergerak (mobile) berbasis Android.
3. Sensor yang digunakan adalah sensor akselerometer.
4. Pemrograman menggunakan bahasa C++ untuk Arduino dan Java untuk bagian antarmuka perangkat bergerak (*mobile*).
5. Pengujian dilakukan oleh sukarelawan dengan berjalan di jalan datar.

1.4. Tujuan

Tujuan pembuatan tugas akhir ini adalah:

1. Mengetahui cara kerja sistem monitoring aktivitas fisik user bergerak dengan analisa langkah (Step Analysis) secara real-

time menggunakan mikrokontroler arduino pada teknologi perangkat bergerak (mobile).
2. Sistem yang dibuat dapat menampilkan kalori tubuh yang terbakar, jarak yang ditempuh, jumlah langkah kaki dan kecepatannya.

1.5. Manfaat

Manfaat dari pembuatan tugas akhir ini antara lain:
1. Mendorong individu untuk bersaing dengan diri mereka sendiri dalam mendapatkan fit dan menurunkan berat badan.
2. Mendapatkan kemudahan dalam mengakses sistem monitoring dimana saja dan kapan saja dengan teknologi perangkat bergerak (*mobile*).

1.6. Metodologi

Adapun langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:
1. Penyusunan proposal tugas akhir

   Pada tahap awal penyusunan proposal, penulis memulai pengerjaan Tugas Akhir dengan melakukan penyusunan Tugas Akhir. Pada proposal tersebut berisi mengenai mekanisme sistem bekerja secara adaptif.

2. Studi literatur

   Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai sensor *accelerometer*, mikrokontroler Arduino, cara mendeteksi langkah kaki, algoritma untuk mengitung jumlah langkah, jarak yang ditempuh, kecepatan dan kalori yang terbakar, Penerimaan dan pengiriman data dari

mikrokontroler Arduino ke *smartphone* kemudian Android sebagai mobile platform dari aplikasi.

3. Analisis dan Desain untuk Perangkat Lunak

   Untuk rancangan dari aplikasi sistem monitoring yang akan dibuat, yaitu:

   1. Sensor accelerometer, mikrokontroler, *bluetooth*, baterai dirangkai menjadi suatu perangkat gadget.
   2. Proses pendeteksian mengukur percepatan atau nilai akselerasi dari 3 sumbu gerakan dilakukan oleh sensor accelerometer, kemudian diolah datanya melalui mikrokontroler.
   3. Perangkat gadget dapat melakukan penghitungan energi yang keluar dari tubuh, langkah kaki dan jarak yang ditempuh.
   4. Perangkat mobile memiliki aplikasi android untuk monitoring nilai total langkah, jarak yang ditempuh, energi yang keluar dari tubuh dari perangkat gadget secara *real-time*.

4. Implementasi perangkat lunak

   Dalam tugas akhir ini nantinya akan dibuat sistem yang dapat melakukan monitor data aktivitas fisik user bergerak. Masukan data awal berupa inisialisasi yaitu tinggi dan berat badan sudah diinputkan di dalam program mikrokontroler dan juga masukan data berupa percepatan atau nilai akselerasi dari 3 sumbu gerakan yang dideteksi oleh sensor akselerometer, yang kemudian diproses oleh mikrokontroler, lalu ditransmisikan ke perangkat *mobile* menggunakan *bluetooth*. Hasil yang ditampilkan di perangkat bergerak (*mobile*) yaitu berupa jumlah energi yang keluar dalam bentuk kalori, jumlah langkah kaki, jarak yang ditempuh dalam bentuk numerik. Berikut beberapa hal yang diperlukan dalam implementasi:

1. IDE untuk perangkat bergerak (mobile) menggunakan Android Studio.
2. Java Runtime Environment.
3. IDE untuk mikrokontroler menggunakan Arduino on Windows.
4. Library tertentu.
5. Mikrokontroler Arduino
6. Modul accelerometer
7. Modul *bluetooth*

5. Pengujian dan evaluasi

   Untuk pengujian dilakukan dengan cara memasukkan data tinggi dan berat badan ke perangkat gadget setelah itu menggunakan perangkat yang sudah menjadi satu rangkaian yang terdiri dari arduino, sensor akselerometer, baterai, *bluetooth* dipasang di *knee pad* pada bagian lutut dengan melakukan aktifitas apa saja, dimana saja dan kapan saja, lalu di monitor dengan perangkat bergerak (*mobile)* berbasis android. Dilakukan proses implementasi dari rancangan yang sudah dibuat untuk menguji apakah aplikasi sistem monitoring sudah dapat menampilkan jumlah langkah, jarak, kecepatan, dan kalori yang terbakar dengan akurat. Data tersebut juga dapat *reset* dari aplikasi android.

6. Penyusunan buku tugas akhir

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu perumusan masalah, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Perangkat Lunak

Bab ini berisi tentang desain sistem yang disajikan dalam bentuk *pseudocode*.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa code yang digunakan untuk proses implementasi

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang telah dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan.

2.1. Sensor Akselerometer (ADXL 345)

ADXL 345 adalah sensor *3-axis accelerometer* beresolusi 13 bit. modul ini Mengukur percepatan statis gravitasi dalam aplikasi tilt-sensing, serta percepatan dinamis yang dihasilkan dari gerakan (*motion*) atau tabrakan (*shock*). resolusi tinggi (4 mg / LSB) memungkinkan pengukuran kemiringan perubahan kurang dari 1 derajat.

Modul ini memiliki 32 bit tipe *First In First Out* (FIFO) yang dapat menyimpan data untuk meminimalisir intervensi mikrokontroler dan menghemat konsumsi energi sistem.

Mode daya rendah memungkinkan manajemen daya hingga terdeteksi gerakan yang melewati ambang batas (*threshold*) berbasis gerak cerdas dengan ambang penginderaan dan pengukuran percepatan aktif pada disipasi daya yang sangat rendah [1].

2.2. Mikrokontroler Arduino Pro Mini

Arduino pro mini seperti yang ditunjukkan pada gambar 2.1 adalah mikrokontroller berbasis ATmega328. Arduino pro mini memiliki 14 pin digital input/output, 6 pin analog, tombol *reset* untuk mengulang program. Arduino pro mini dirancang untuk kebutuhan semi permanen karena tidak memiliki pin header, sehingga memungkinkan pengguna untuk menyambung dengan berbagai macam konektor maupun solder langsung dari

kabel. Arduino pro mini ini beroperasi dengan tegangan 3.3 V dan 8 MHz. Chip ATmega328 pada Arduino Pro Mini memiliki memori 32 KB, dengan 0.5 KB dari memori tersebut telah digunakan untuk bootloader. Jumlah SRAM 2 KB, dan EEPROM 1 KB [2].

2.3. EEPROM

EEPROM digunakan untuk menyimpan sebuah nilai, EEPROM bersifat non volatile, dimana data yang tersimpan tidak hilang meski power supply dimatikan.

Pada arduino pro mini menggunakan ATMega328 yang memiliki EEPROM sebesar 1 Kb, yang terdiri dari 1024 bytes, dimana 1 bytes hanya terdiri dari 8 bit. Karena lebar data hanya 8 bit maka nilai data yang diampung adalah 0 hingga 255.

Menulis di EEPROM menghabiskan waktu 3,3 ms. Penyimpanan data di EEPROM hanya dapat menulis data untuk 100000 kali, Namun untuk membaca data dapat dilakukan hingga tak terbatas [3].

2.4. *Bluetooth Module* (HC-06)

Bluetooth HC-06 seperti yang ditunjukkan pada gambar 3 adalah modul komunikasi nirkabel bertipe 2.0 dengan koneksi hanya sebagai *SLAVE*. Interface yang digunakan adalah serial RXD, TXD, VCC dan GND. Built in LED sebagai indikator koneksi *bluetooth*. Modul ini berkerja pada tegangan 3.3 V DC. Arus saat *unpaired* sekitar 25mA, dan saat *paired* (terhubung) sebesar 8mA. Modul ini tidak dapat mode tidur (*sleep mode*). Jarak efektif jangkauan sebesar kurang dari 20 meter. [4].

2.5. Mendeteksi langkah kaki

Metode Antara Koordinat X, Y dan Z pilih yang memiliki perubahan percepatan koordinat axis paling besar.

Setiap langkah terhitung jika ada kurva dari koordinat percepatan ketika kurva melewati garis batas threshold.

Dynamic threshold level diambil dari (*max* + *min*)/2. Setiap *threshold* diperbarui setiap 2 detik, dimana dalam 2 detik bisa memproses 13 sampel pendeteksian nilai akselerasi, jadi *threshold* bergerak dinamis. Nilai max dan min diambil dari 13 sampel tersebut. level *threshold* ini digunakan untuk memutuskan apakah langkah sudah terhitung. karena sistem ini update setiap 13 sampel, *threshold* bergerak dinamis. Pilihan ini adaptif dan cukup cepat, sebagai tambahan terhadap *dynamic threshold*, *dynamic precision* juga digunakan untuk *filtering* [5].

Menghitung langkah kaki dapat bekerja dengan baik dengan menggunakan algoritma ini, tapi terkadang bisa menjadi terlalu sensitif. Ketika *gadget* ini bergetar sangat lambat dari penyebab selain berjalan atau berlari. Getaran yang tidak valid tersebut harus dibuang supaya dapat menentukan ritme langkah yang benar. Peraturan tambahan digunakan untuk memecahkan masalah ini.

Peraturan tambahan ini unuk menyeleksi penyeleksian ketika kurva melewati *threshold* sudah masuk kategori bergerak atau tidak bergerak. Dengan cara memberi standar nilai bergerak yaitu nilai jarak antara nilai kurva tertinggi "bukit" dengan *threshold* atau nilai kurva terendah "lembah" dengan *threshold*.

2.6. Menghitung Jarak

Dari algoritma mendeteksi langkah kaki, dapat diketahui jumlah langkah kaki, Dengan mengetahui jumlah langkah kaki dan panjang setiap langkah kaki, maka menghitung jarak yang ditempuh diketahui dengan rumus

Jarak = jumlah langkah kaki * panjang setiap langkah

Tabel 2.1. Tabel Hubungan Jumlah Langkah per 2 detik dengan stride

Jumlah Langkah kaki per 2 detik	*stride* (m/s)
0~2	Tinggi Badan/5
2~3	Tinggi Badan/4
3~4	Tinggi Badan/3
4~5	Tinggi Badan/2
5~6	Tinggi Badan/1.2
6~7	Tinggi Badan
>=8	1.2 * Tinggi badan

2.7. Menghitung Kecepatan

Dengan mengetahui jarak yang ditempuh dan waktu maka dapat diperoleh kecepatannya. Dari jumlah langkah kaki per 2s dan stride, maka menghitung kecepatan diketahui dengan rumus

Kecepatan = (langkah kaki per 2 detik * *Stride*)/2

2.8. Menghitung Nilai Kalori yang Terbakar

Tidak ada cara akurat untuk menghitung kalori. Beberapa faktor seperti berat badan, intensitas aktifitas, kondisi tubuh dan metabolisme. Untuk mengkalkulasi kalori menggunakan cara konvensional. Menghitung kalori diketahui dengan rumus:

Kalori(C/kg/h) = 1.25 * kecepatan

Tapi, pada proposal ini satuan kecepatan yang digunakan adalah m/s maka diketahui dengan rumus:

Kalori(C/kg/h) = 1.25 * kecepatan
Kalori(C/kg/h) = 1.25 * kecepatan * 3600/1000
= 4.5 * kecepatan

Dengan mempertimbangkan berat badan. Maka diketahui dengan rumus:

Kalori (C/2s) = 4.5 * kecepatan (m/s) * berat badan/1800

Jika pengguna sedang beristirahat, maka tidak ada perubahan pada jumlah langkah kaki dan jarak yang ditempuh, kecepatan menjadi 0. Keluaran kalori saat istirahat berkisar pada 1 C/kg/hour. Maka diketahui dengan rumus [5].

Kalori (C/2s) = 1 * berat badan/1800

2.9. Perangkat bergerak Android/*mobile*

Android merupakan sistem operasi *open source*. Sebagai OS *open source*, semua pihak dapat memodifikasi perangkat lunak tanpa khawatir adanya hak paten dan didistribusikan oleh para pembuat perangkat, operator nirkabel, dan pengembang aplikasi. Android umumnya ditulis dalam bahasa pemrograman Java [6].

2.10. Arduino *to* Java

Arduino pro mini adalah sebuah jenis mikrokontroler yang menggunakan bahasa C dalam pemrogramannya. Namun, melalui *serial port*, Arduino dapat menjadi fleksibel dengan beberapa bahasa pemrograman lainnya, salah satunya Java. Arduino dapat menampilkan data dengan cara berkomunikasi dengan *serial port*.

[Halaman ini sengaja dikosongkan]

BAB III
PERANCANGAN PERANGKAT LUNAK

Perancangan merupakan bagian penting dari pembuatan perangkat lunak yang berupa perencanaan-perencanaan secara teknis aplikasi yang dibuat. Sehingga bab ini secara khusus akan menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir ini. Berawal dari deskripsi umum aplikasi hingga perancangan proses, alur dan implementasinya.

3.1. Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun suatu sistem monitoring Aktivitas fisik user bergerak dengan analisa langkah (*step analysis*) untuk estimasi pembakaran kalori dengan menggunakan mikrokontroler Arduino. Aplikasi monitoring Aktivitas fisik user bergerak dengan studi kasus pada seseorang yang bergerak dirancang untuk melakukan pemantauan jumlah kalori yang terbakar, jarak yang dia tempuh dan total jumlah langkah yang ditempuh selama bergerak. Data yang dibutuhkan untuk menentukan langkah, yaitu data percepatan sudut akselerometer.

Dibutuhkan pula satu jenis sensor dalam perancangan arsitektur sistem ini, yaitu sensor akselerometer pada Arduino. Perancangan akan dibagi menjadi tiga proses utama, yaitu:

1. Perancangan mikrokontroler Arduino dengan integrasi sensor akselerometer untuk menangkap data akselerasi pada pergerakan kaki.
2. Perancangan pengolahan data akselerasi pada mikrokontroler Arduino untuk menghasilkan total langkah, jarak yang ditempuh dan kalori yang terbakar.
3. Perancangan aplikasi monitoring untuk memonitor data total langkah, jarak yang ditempuh dan kalori yang terbakar yang diolah pada perangkat mikrokontroler.

3.2. Arsitektur Umum Sistem

Pada bab ini akan dijelaskan perancangan alat dan program yang dibuat. Perancangan akan dibagi menjadi dua proses utama, yaitu:

1. Perancangan mikrokontroler Arduino dengan integrasi sensor akselerometer untuk menangkap data percepatan sudut atau nilai akselerasi pada kaki yang bergerak.
2. Perancangan aplikasi untuk pengolahan data sensor akselerometer (nili akselerasi atau percepatan sudut) menjadi nilai total jumlah langkah, total jarak yang ditempuh dan total kalori yang terbakar yang ditanam mikrokontroler.
3. Perancangan aplikasi monitoring untuk memonitor nilai total langkah, total jarak yang ditempuh dan total kalori yang terbakar yang didapat dari mikrokontroler. Dan fungsi untuk *reset data* mikrokontroler menjadi 0

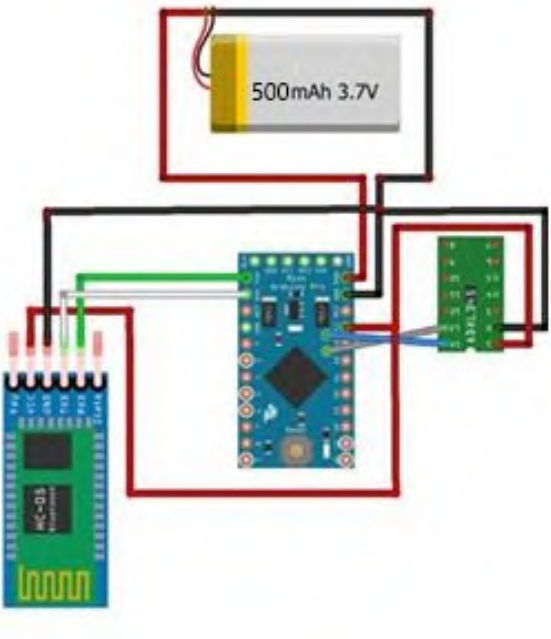
Pada bab ini juga akan dijelaskan mengenai gambaran dan alur kerja sistem secara umum.

3.3. Perancangan Perangkat Keras

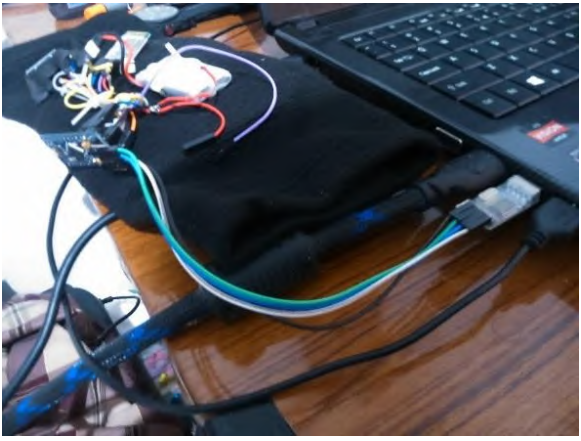
Pada sistem ini, perangkat keras yang digunakan sebagai berikut:

1. Satu buah mikrokontroler Arduino Pro Mini 3.3v,
2. Satu buah sensor Akselerometer ADXL 345,
3. Satu buah *bluetooth module* HC-06,
4. Satu buah *USB to TTL* PL2303HX,
5. Satu set kabel *jumper*,
6. Satu buah baterai *Li po* 3.7V 500mah,
7. Satu buah HP Android Sony Xperia L,

Rangkaian perangkat mikrokontroler ditunjukkan pada Gambar 3.1 dan Gambar 3.2.



Gambar 3.1. Rancangan perangkat keras



Gambar 3.2. Kabel USB to TTL

Perangkat keras dirangkai menggunakan solder. Mikrokontroler, perangkat sensor, dan *bluetooth module* dihubungkan satu sama lain dengan rangkaian yang sudah diolder. Pada rangkaian di atas, mikrokontroler berfungsi sebagai "otak" dari perangkat sensor yang akan mengontrol kerja perangkat sensor. Program diunggah di mikrokontroler ini. Untuk mengunggah program ke dalam mikrokontroler, dibutuhkan suatu USB to TTL yang menghubungkan antara Laptop dan rangkaian perangkat keras pada Gambar 3.2.

Komponen lainnya adalah sensor Akselerometer yang bertugas untuk menangkap nilai akselerasi yang ada di objek berupa satuan tertentu. Akselerasi dikalibrasi ke dalam satuan *meter per second squared* (m/s^2). Selain itu, pada rangkaian diatas terdapat *EEPROM* di dalam arduino yang berfungsi untuk penyimpanan file hasil penyimpanan data dengan kapasitas 1 Kb. Perangkat keras ini dapat berjalan ketika ada *power* yang mengisinya, dengan menggunakan baterai *li-po*.

3.4. Perancangan Integrasi Mikrokontroler dan Komponen Sensor

Sensor akselerometer merupakan salah satu alat utama pada sistem. Sensor tersebut akan digunakan untuk menangkap data akselerasi perubahan pergerakan pada kaki. Data akselerasi diolah oleh mikrokontroler arduino untuk menghasilkan output jumlah langkah, jarak yang ditempuh, dan kalori yang terbakar. Untuk mengirim data total langkah, total jarak yang ditempuh dan total kalori yang terbakar ke aplikasi android di handphone maka diperlukan modul transmisi data yaitu modul *bluetooth*. Gambaran umum perancangan alat dapat dilihat pada Gambar 3.1.

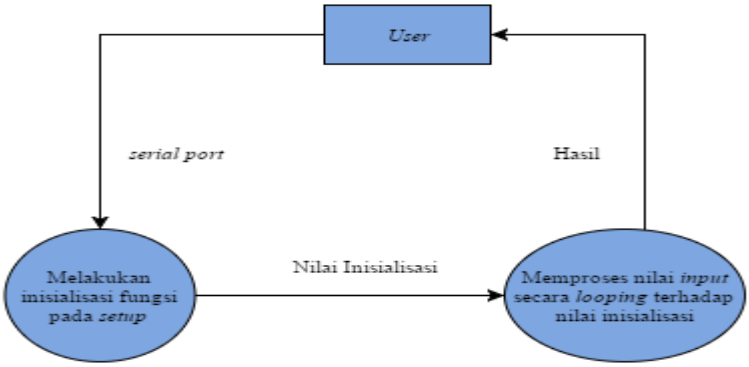
3.5. Diagram Alir Aplikasi Sistem

Alur setiap proses yang terdapat pada aplikasi digambarkan pada diagram alir, untuk memudahkan

pemahaman secara garis besar proses yang ada pada sistem. Diagram alir aplikasi sistem terdiri dari enam proses utama, yaitu mendeteksi nilai akselerometer, menghitung langkah, menghitung *stride*, menghitung jarak, dan menghitung kalori yang terbakar. Pada subbab ini, terdapat pula diagram alir menjalankan aplikasi android.

3.5.1. Diagram Alir Inisialisasi Arduino

Ketika Arduino dinyalakan pertama kali, Arduino akan melakukan inisialisasi pada fungsi **setup**. Apapun yang diinisialisasikan pada fungsi **setup** ini hanya dilakukan sekali ketika Arduino pertama kali dijalankan. Pada proses tersebut, Arduino akan mengatur *setting serial port.* Inisialisasi akan digunakan untuk proses pendeteksian serta penentuan kondisi udara pada proses selanjutnya. Diagram alir inisialisasi Arduino ditunjukkan pada Gambar 3.3.

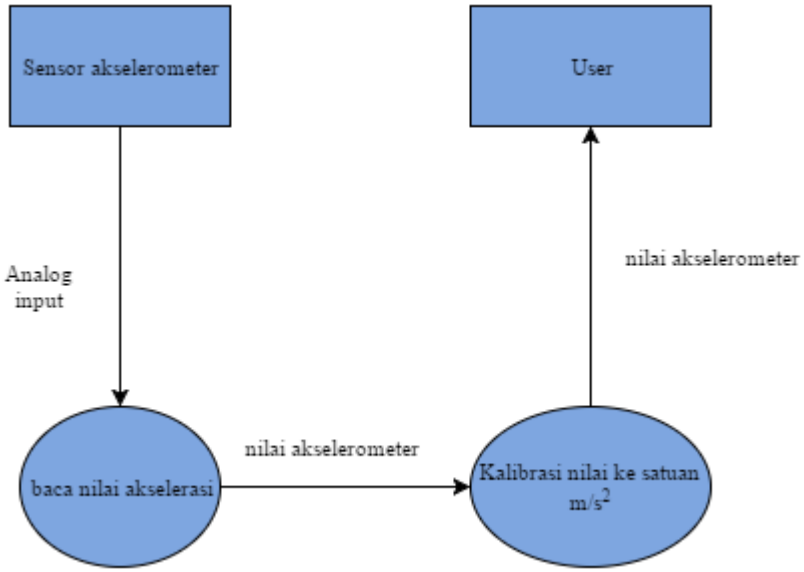


Gambar 3.3. Diagram alir inisialisasi Arduino

3.5.2. Diagram Alir Mendeteksi Nilai Sensor Akselerometer (ADXL345)

Nilai *input* akselerasi yang diperoleh dari *analog pin* perangkat sensor memiliki nilai berupa data bit. Ketika nilai

sensor diperoleh, data tersebut harus dikalibrasikan terlebih dahulu jika ingin mendapatkan data dalam satuan m/s^2 (*meter per second squared).* Data yang diterima disimpan dalam *array*. Pembacaan data dapat dilakukan secara berulang dengan *delay* waktu tertentu. Diagram alir untuk mendeteksi nilai akselerasi ditunjukkan pada Gambar 3.4.



Gambar 3.4. Diagram alir mendeteksi nilai sensor

3.5.3. Diagram Alir Menghitung Langkah

Data nilai akselerometer diambil selama 2 detik. Selama 2 detik, jumlah data yang terambil adalah 54 nilai, dengan masing-masing sudut mendapat 18 nilai. kumpulan nilai tersebut disimpan ke dalam 3 array yang berbeda, 3 *array* tersebut menandakan 3 sudut akselerometer, yaitu x,y dan z.

Dari 3 sudut akselerometer, diambil salah satu untuk menentukan sudut teraktif. Nilai akselerometer pada sudut teraktif inilah yang menjadi dasar untuk deteksi langkah. Untuk

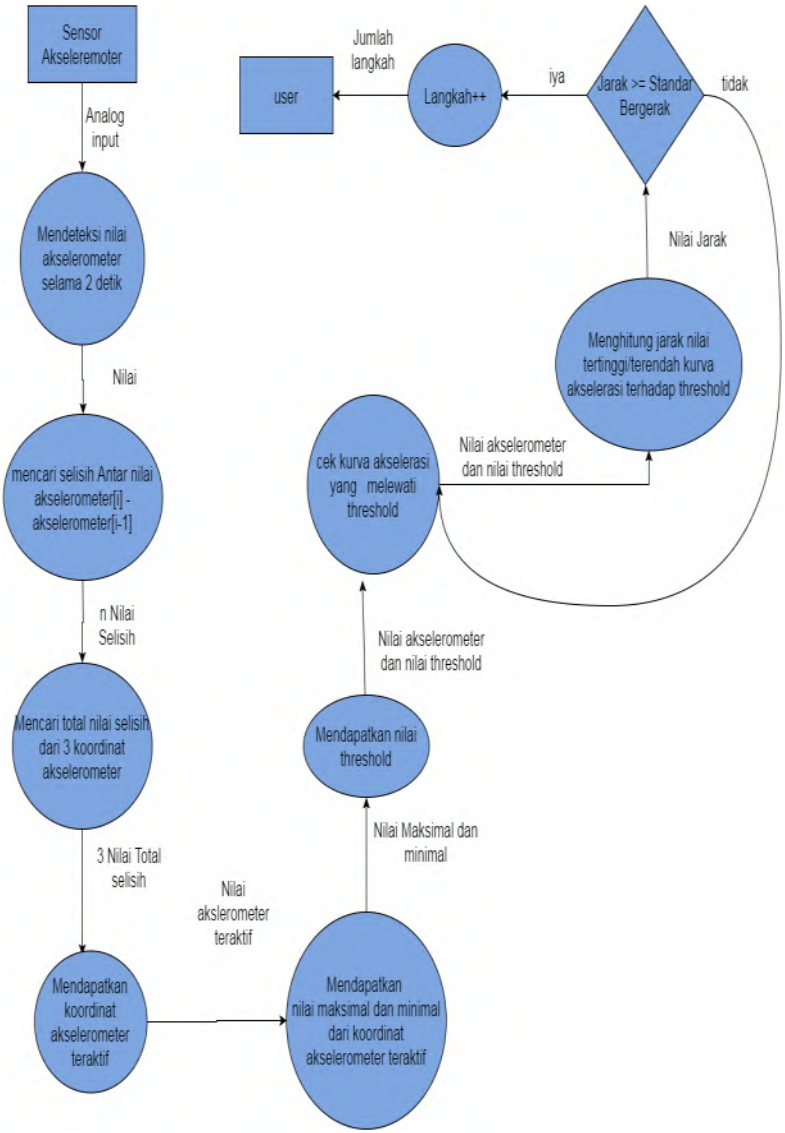
menentukan sudut teraktif diawali dengan mendapatkan selisih antar 18 data nilai akselerasi yang tersimpan di *array* pada masing-masing akselerometer. Setelah didapatkan nilai selisih lalu nilai selisih itu di totalkan hingga 17 kali. Total nilai selisih dari 3 sudut akselerometer diurutkan menggunakan *bubble sort* untuk diambil nilai yang terbesar untuk mendapatkan sudut akselerometer yang teraktif.

Pada sudut akselerometer teraktif, dari 18 data nilai akselerasi, diambil nilai maksimal dan minimal. Nilai maksimal dan minimal ini digunakan untuk menghitung nilai *threshold*[5]. *Threshold* diperoleh menggunakan rumus*:*

$$Threshold = (\text{maksimal} + \text{minimal})/2$$

Setelah didapatkan nilai *threshold*, maka nilai akselerasi dicek untuk menentukan apakah pergerakan nilai akselerasi telah melintasi *threshold* dengan cara mengecek nilai akselerasi sudah melewati *threshold* dan satu nilai akselerasi sebelumnya belum melewati *threshold*, pengecekan dua nilai akselerasi tersebut yang menjadikan pergerakan nilai akselerasi yang melintasi *threshold* dilakukan hingga nilai akselerasi terakhir pada sudut teraktif tersebut.

Setelah terjadi proses ketika kedua nilai akselerasi menunjukkan pergerakan akselerometer telah melintasi *threshold*, dilakukan sebuah aturan untuk pengkategorian nilai akselerometer itu termasuk kategori bergerak atau tidak bergerak. Pengkategorian bergerak dengan cara menghitung jarak antara *threshold* dengan nilai terendah apabila nilai sebelum lintasan lebih kecil dari *threshold* dan jarak antara *threshold* dengan nilai tertinggi apabila nilai sebelum lintasan lebih besar dari *threshold*. Ketika jarak antara nilai akselerasi yang tertinggi/terendah lebih besar dari nilai standar bergerak yang sudah ditentukan, maka itu sudah terjadi suatu langkah. Diagram alir menghitung langkah ditunjukan pada Gambar 3.5.



Gambar 3.5. Diagram alir Menghitung Langkah

3.5.4. Algoritma Menghitung *Stride*

Stride adalah panjang dari tumit ke tumit dalam setiap langkah. untuk memperoleh stride diperoleh dari pengolahan data jumlah langkah setiap 2 detik dan tinggi badan menggunakan metode *fuzzy*[5]. Nilai tinggi badan didapatkan dengan inisialisasi yang ditanam pada arduino. Dimana *fuzzy* dijelaskan pada tabel 3.1.

Tabel 3.1. Tabel Hubungan Jumlah Langkah per 2 detik dengan stride

Langkah per 2 detik	***Stride*(m/s)**
0~2	Tinggi Badan/5
2~3	Tinggi Badan/4
3~4	Tinggi Badan/3
4~5	Tinggi Badan/2
5~6	Tinggi Badan/1.2
6~8	Tinggi Badan
>=8	1.2 x Tinggi Badan

3.5.5. Algoritma Menghitung Jarak

Pada rumus menghiung jarak adalah kecepatan dikalikan dengan waktu proses per ***loop*** mikrokontroler, Untuk itu maka harus diketahui berapa nilai kecepatannya[5], kecepatan diperoleh menggunakan rumus:

Kecepatan = (Langkah per 2 detik * *stride*)/2

Setelah mendapat nilai kecepatan maka bisa dicari nilai jaraknya, yaitu dengan mengkalikan nilai kecepatan dengan waktu[7]. Waktu pada algoritma ini adalah waktu proses setiap ***loop*** mikrokontroler yaitu 2 detik, Pengukuran jarak diketahui menggunakan rumus:

Jarak = Kecepatan * 2 detik

3.5.6. Algoritma Menghitung Kalori

Perhitungan kalori terbagi menjadi 2 klasifikasi, yaitu klasifikasi bergerak dan tidak bergerak, perhitungan kalori dikalibrasi ke dalam satuan calories per kilogram per hour (c/kg/h) namun karena kalori dihitung per 2 detik maka satuan menjadi *calories per 2 seconds*, Untuk klasifikasi bergerak, diperlukan inisialisasi berat badan dan kecepatan. Nilai berat badan didapatkan dengan inisialisasi yang ditanam pada arduino[5]. Didapati kalori yang terbakar di klasifikasi bergerak dengan rumus:

Calories (c/kg/h) = 1.25 * Kecepatan * 3600/1000

= 4.5 * Kecepatan

Calories (C/2 s) = 4.5 * Kecepatan * Berat Badan/1800

= Kecepatan * weight/400

Pada saat termasuk dalam kondisi tidak bergerak, maka tidak ada perubahan dalam melangkah, jarak yang ditempuh dan kecepatan. Dengan menggunakan rumus:

Calories (C/2 s) = 1 * Berat Badan/1800

Pada akhirnya seluruh kalori setiap 2 detik akan ditotalkan.

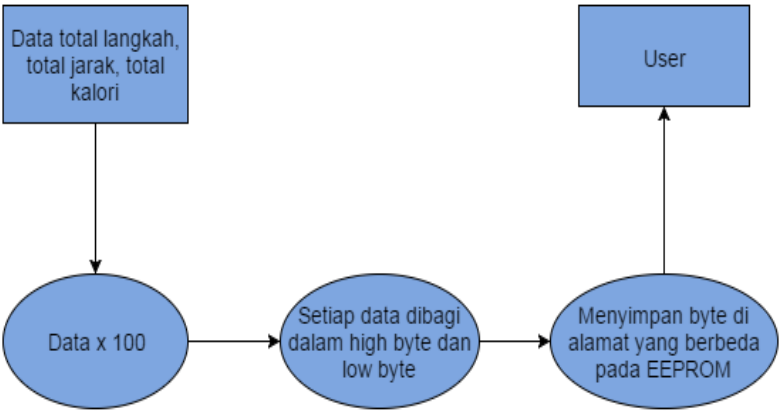
3.5.7. Perancangan Penyimpanan Data di EEPROM

Data total langkah, total jarak yang ditempuh dan total kalori yang diggunakan disimpan ke dalam EEPROM. dikarenakan keterbatasan EEPROM hanya bisa menyimpan

data antara 0 - 255 dan bilangan koma juga tidak bisa disimpan, maka data dikali 100 supaya tidak menjadi bilangan koma, data disimpan dengan *byte* data dibagi menjadi *high byte* dan *low byte*, cara pembagian ini sudah tersedia pada libary EEPROM. Diagram alir penyimpanan data di EEPROM dapat dilihar di Gambar 3.6. Didapati rumus untuk penyimpanan nilai data *high byte* dan *low byte* adalah sebagai berikut:

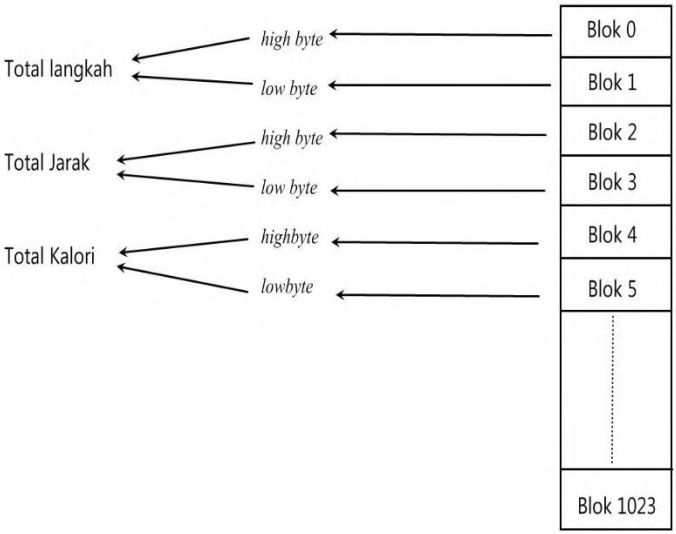
High byte = Nilai / 256

Low byte = Nilai % 256

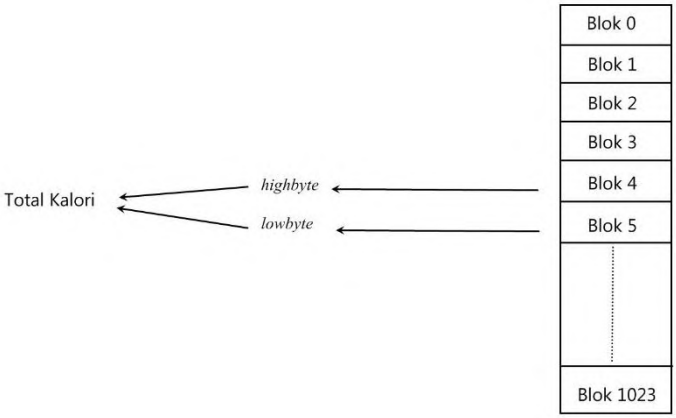


Gambar 3.6. Diagram alir Menyimpan di EEPROM

Untuk penyimpanan saat terjadi perubahan nilai total langkah dan total jarak maka setiap terdapat nilai data baru, nilai pada setiap alamat yang sebelumnya menjadi tempat penyimpanan akan diperbarui, sistem penyimpanan didalam *storage* dapat dilihat pada Gambar 3.7. Sedangkan, ketika total langkah dan total jarak tidak terjadi perubahan maka nilai yang disimpan hanya total kalori, sistem peyimpanan didalam *storage* dapat dilihat pada gambar 3.8.



Gambar 3.7. Sistem Penyimpanan Data di Dalam Storage Pada Kondisi Terjadi Perubahan Nilai Total Langkah dan Total Jarak



Gambar 3. 8 Sistem Penyimpanan Data di Dalam Storage pada Kondisi Diam

3.6. Rancangan Antar Muka Sistem

Pengguna dapat menjalankan 4 fitur utama, yaitu mengaktifkan *bluetooth*, melakukan pair dengan *bluetooth* mikrokontroler, monitoring data dari mikrokontroller dan melakukan reset data ke 0. Fitur mengaktifkan *bluetooth*, melakukan pair dan memonitor data akan dirancang antar muka yang sederhana dan mudah dipahami oleh pengguna. Sedangkan fitur melakukan reset data merupakan fitur perintah. Khusus untuk antarmuka fitur mengaktifkan *bluetooth* hanya muncul apabila *bluetooth* belum diaktifkan. Rancangan antarmuka untuk fitur mengaktifkan *bluetooth* dapat dilihat pada Gambar 3.9.

Gambar 3.9. Rancangan Antarmuka untuk pengaktifan Bluetooth

Gambar 3.9 merupakan rancangan antar muka aplikasi android bagian mengaktifkan *bluetooth*. Untuk fitur mengaktifkan *bluetooth* akan muncul 2 tombol pilihan, yaitu

"Allow" untuk mengaktifkan *bluetooth*, dan "Deny" untuk tidak mengaktifkan *bluetooth*. Apabila sudah masuk ke pengaktifan *bluetooth* maka akan muncul antarmuka untuk fitur melakukan pair dengan *bluetooth* mikrokontroler dapat dilihat pada Gambar 3.10

Gambar 3.10. Rancangan Antarmuka untuk memilih pair bluetooth

Gambar 3.10 merupakan rancangan antar muka aplikasi android bagian melakukan pair dengan *bluetooth*. Untuk fitur melakukan pair dengan *bluetooth* akan menampilkan *list bluetooth* yang sudah pernah di *pair* oleh handphone, dimana user hanya perlu memilih salah satu untuk nama *bluetooth*, pada gambar 3.10 terdapat 3 contoh pilihan yaitu HC-06, CROSS A7S, Fauzi. setiap pilihan terdapat 2 baris *text*,baris atas adalah nama *bluetooth* dan baris bawah adalah MAC address

bluetooth. Setelah memilih *bluetooth* akan menampilkan antar muka bagian memonitor data dari mikrokontroler.

Gambar 3.11. Rancangan Antarmuka untuk mendapat nilai langkah, jarak dan kalori

Gambar 3.11 merupakan rancangan antar muka aplikasi android bagian memonitor data dari mikrokontroler. Untuk menampilkan data dari mikrokontroler akan otomatis muncul pada *text* yang terletak dibawah *button reset* sedangkan untuk melakukan reset data ditunjukan pada *button* bertuliskan *reset*.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi sistem pada perangkat keras dan perangkat lunak dari perancangan sistem yang telah dibahas pada bab 3.

4.1. Lingkungan Implementasi

Dalam merancang perangkat lunak ini digunakan beberapa perangkat pendukung sebagai berikut.

4.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem adalah komputer, mikrokontroler Arduino dan handphone. Spesifikasi dari perangkat-perangkat tersebut adalah sebagai berikut:

1. Laptop Acer e451g, AMD 4500m 1.9 GHz dan 8 GB memori DDR3,
2. Arduino pro mini ATmega328P dengan *clock speed 8 MHz* (3.3V model) 32 *Kb flash* memori, 2 *Kb* memori SRAM, 1 Kb EEPROM.
3. Hand phone Sony Xperia L dengan 1 GB ram.

4.1.2. Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan sistem adalah sebagai berikut:

1. Microsoft Windows 7 sebagai sistem operasi,
2. Android Stuudio untuk mengimplementasikan aplikasi android,
3. Arduino *Development Kit* versi 1.6.8 sebagai IDE untuk mengimplementasikan aplikasi Arduino,
4. Website *draw.io* untuk merancang diagram alir data.

4.2. Implementasi Perangkat Keras

Pembuatan implementasi perangkat keras ini diawali dengan sebuah *prototype* untuk menguji apakah perangkat keras dapat berfungsi sebagai pengambilan data sensor. Pada sistem ini, perangkat keras yang digunakan sebagai berikut:

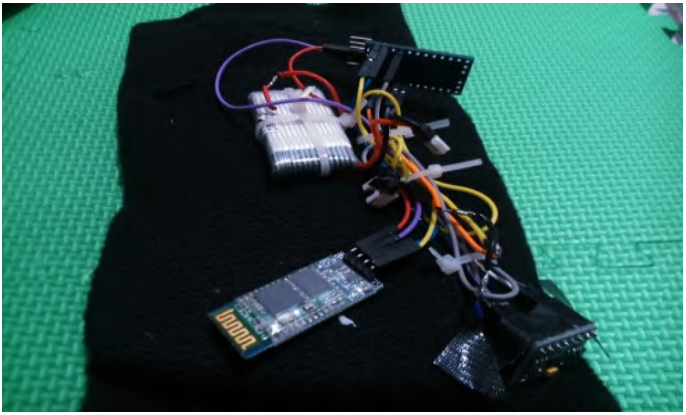
1. Satu buah mikrokontroler Arduino pro mini *8 MHz 3.3v*,
2. Satu buah sensor akselerometer ADXL345,
3. Satu buah *bluetooth module* HC-06,
4. Satu buah baterai *li-po 3.7v 500 mah*,
5. Satu buah USB to TTL PL2303HX,
6. Satu buah solder 60w,
7. Satu set timah solder,
8. Satu set kabel,
9. Satu buah Knee pad,
10. Satu buah kabel USB,
11. Satu buah Handphone android Sony Xperia L,
12. Satu buah laptop Acer E1-451G.

Pengujian fungsi perangkat sensor melibatkan semua komponen perangkat. *Prototype* perangkat keras pada tugas akhir ini menggunakan sebuah mikrokontroler Arduino yang dilengkapi dengan sebuah *bluetooth module* dan satu buah sensor, yaitu sensor akselerometer ADXL 345. Mikrokontroler Arduino berfungsi sebagai "otak" dari perangkat yang melakukan kontrol algoritma dan pemrosesan data.

Arduino ditanami program oleh laptop dengan cara menyambungkan arduino dengan laptop menggunakan USB to TTL(PL2303HX).

Untuk menghubungkan mikrokontroler dengan sensor, *bluetooth* dan baterai *li po*, dihubungkan dengan solder. Pada sensor akselerometer (ADXL 345) terdapat empat *pin* utama agar sensor dapat berjalan. Keempat *pin* tersebut adalah *vcc, GND, analog pin 4* dan *analog pin 5*. Nilai konsentrasi yang

diperoleh berasal dari *analog pin*. Pada *bluetooth module*(HC-06) terdapat empat *pin* utama agar *bluetooth* dapat berjalan. Keempat *pin* tersebut adalah *vcc, GND, digital pin 0(RX)* dan *digital pin 1(TX)*. Pada baterai(*li-po* 3.7v 500 mah) terdapat dua *pin* utama agar baterai dapat memberi energi. Kedua *pin* tersebut adalah *raw* dan GND. Baterai menggunakan *pin* bagian *raw*, karena voltase yang diterima bukan 3.3v, untuk 3.3v listrik dipasok ke *pin* bagian *vcc*. Gambar rangkaian perangkat sensor ditunjukkan pada Gambar 4.1.

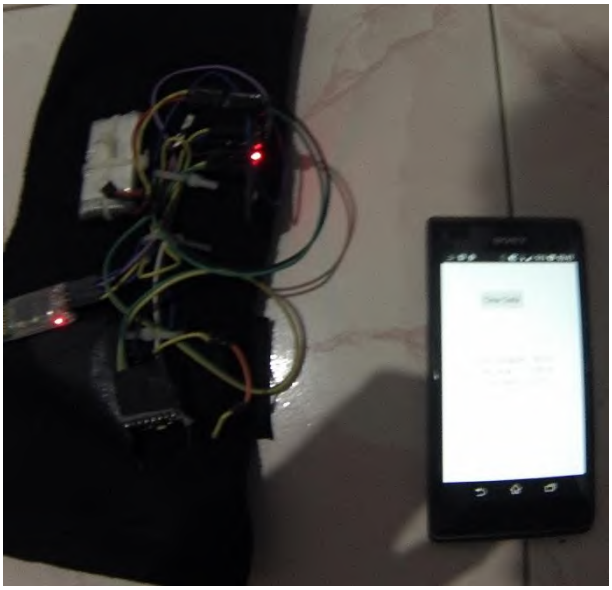


Gambar 4.1. Rangkaian perangkat keras

Perangkat sensor pada Gambar 4.1. dapat berjalan jika ada *power* yang mendukung perangkat tersebut. *Power* dapat diperoleh dari baterai *li-po 3.7v 500mah* langsung. *Power* yang diperoleh digunakan oleh sensor daan *bluetooth* melalui *vcc pin*. *Power* yang digunakan beragam. Untuk aplikasi ini, *power* yang digunakan sebesar 3.7v.

Percobaan awal mikrokontroler yang sudah terhubung dengan sensor, *bluetooth* dan *li-po* baterai di pasang di lutut menggunakan knee pad untuk mendapat nilai akselerasi dan pengolahan data. Pada penyimpanan data prototype ini, EEPROM internal arduino bermain peran sebagai penyimpan data hasil total langkah total jarak dan total kalori, selain itu juga

untuk melakukan *reset* data menjadi 0 pada EEPROM. Pengguna perangkat ini bergerak secara berjalan dan istirahat. Agar nilai pada perangkat mikrokontroler dapat ditransmisi dengan aplikasi di handphone android, diperlukan suatu *bluetooth* untuk menghubungkannya. Spesifikasi untuk handphone yang dapat digunakan adalah yang memiliki internal *bluetooth* dan untuk perangkat mikrokontroler menggunakan *bluetooth* module HC-06. Gambar perangkat mikrokontroler yang tersambung dengan aplikasi android pada handphone menggunakan *bluetooth* ditunjukkan pada Gambar 4.2.



Gambar 4.2. Rangkaian perangkat keras tersambung dengan aplikasi android

4.2.1. Anggaran Implementasi Perangkat keras

Seluruh peralatan untuk implementasi perangkat keras untuk membangun perangkat ini menghabiskan biaya yang tertera pada Tabel 4.1.

Tabel 4. 1. Anggaran biaya implementasi perangkat keras

No.	Uraian	Unit/ Satuan	Volume	Harga Satuan (Rp)	Jumlah (Rp)
1	Arduino Pro Mini 8 MHz 3.3v	Unit	1	45.000,00	Rp 45.000,00
2	Sensor akselerometer ADXL 345	Unit	1	55.000,00	Rp 55.000,00
3	Bluetooth module HC-06	Unit	1	105.000,00	Rp 105.000,00
4	li-po *battery* 3.7v 500 mah	Unit	1	39.500,00	Rp 39.500,00
5	USB to TTL PL2303HX	Unit	1	9.500,00	Rp 9.500,00
6	Solder 60w	Unit	1	12.500,00	Rp 12.500,00
7	Timah Solder	Unit	1	12.000,00	Rp 12.000,00
8	kabel female to female	Unit	12	1.000,00	Rp 12.000,00
9	kabel female to male	Unit	4	1.000,00	Rp 4.000,00
10	Knee pad	Unit	1	32.500,00	Rp 32.500,00
11	Kabel USB	Unit	1	2.500,00	Rp 2.500,00
Jumlah Biaya					**Rp 329.500,00**

4.3. Implementasi pada Mikrokontroler Arduino

Implementasi pada mikrokontroler arduino tentang program yang ditanam pada mikrokontroler Arduino. Setiap bagian akan dijelaskan lebih lanjut pada masing-masing subbab.

4.3.1. Inisialisai Arduino

Proses inisialisasi pada Arduino merupakan suatu hal penting. Inisialisasi dilakukan hanya di awal ketika perangkat sensor pertama kali dinyalakan. Inisialisasi Arduino seperti yang ditunjukkan pada Gambar 4.3.

Inisialisasi arduino dapat dijelaskan melalui beberapa tahapan sebagai berikut:

1. Buka *serial port* dan inisialisasi *baud rate* menjadi 9600.
2. Inisialisasi untuk memilih ukuran mendeteksi dengan jangkauan 16g (16 x 9.81 m/s$^{2)}$.

```
1  Set baud rate to 9600
```

Gambar 4.3. Inisialisasi Arduino serial port

4.3.2. Mendeteksi Nilai Sensor

Terdapat satu jenis sensor yang digunakan pada perangkat keras ini, yaitu sensor akselerometer. Gambar 4.4 dijelaskan mengenai tahapan menangkap nilai akselerometer. Tahapan dalam menangkap nilai sensor dijelaskan sebagai berikut:

1. Implementasi ini berjalan pada sebuah fungsi *loop* dengan parameter percepatan dalam m/s^2 (*meter per second squared)* untuk mengambil satu nilai konsentrasi akselerometer.
2. Proses pendeteksian nilai akselerasi sudut x,y dan z dilakukan selama 54 kali, dimana setiap sudut mendapat 18 nilai karena perhitungan langkah dilakukan setiap 2 detik.
3. Inisialisasi fungsi *event* untuk pengambilan nilai akselerasi, fungsi *event* ini tersedia pada library ADXL 345.
4. Nilai akselerasi pada masing-masing sudut (x, y, z) disimpan ke dalam *array* yang masing-masing berjumlah 18.

```
1  for n ← 0 to 18
2   set event to accelerometer event
3   x[n] ← event.acceleration.x
4   y[n] ← event.acceleration.y
5   z[n] ← event.acceleration.z
```

Gambar 4.4. Mendeteksi nilai sensor akselerometer

4.3.3. Menentukan Koordinat Akselerometer Teraktif

Dalam menentukan koordinat teraktif ini diperoleh dari total selisih 18 nilai data akselerometer ini, dimana selisihnya dari nilai saat ini dengan nilai sebelumnya. Pada Gambar 4.5 ditunjukan tahapan dalam menentukan akselerometer teraktif. Proses menentukan akselerometer teraktif yang dapat dijelaskan sebagai berikut:

1. Proses pengambilan data akselerasi dari *array* koordinat x,y dan z dilakukan selama 18 kali.
2. Mengambil data nilai akselerasi ke-n dan ke n-1 untuk dihitung selisih nilai kedua data tersebut.
3. Bila hasil selisih bernilai negatif, maka dirubah ke positif dengan dikali dengan -1.
4. Menjumlahkan nilai selisih dengan menghitung total nilai selisih yang sudah ada dengan nilai selisih yang baru di diperoleh.
5. Memasukkan 3 total selisih akselerometer (x,y,z) ke dalam *array* urut.
6. Melakukan pengurutan total selisih dari 3 koordinat akselerometer (x,y,z) dari *array* urut dengan menggunakan algoritma *bubble sort.* Algoritma *bubble sort* ditunjukan pada Gambar 4.6.

```
1  Retrieve accelerometer input
2  selisih      ←      sudutAccelerometer[i]      -
   sudutAccelerometer[n-i]
3  if selisih < 0 then
4    selisih * -1
5  total selisih ← total selisih + selisih
6  urut[3] ← total selisih(sudut x, y,z)
7  sort ← urut[3]
8  sudut teraktif = urut[2]
```

Gambar 4.5. Menentukan koordinat akselerometer teraktif

```
1  void function sort
2   initialize (total_selisih, size)
3   for i ← 0 to size-1
```

```
4     for o ← 0 to size − (i+1)
5       if total_selisih[o] > total_selisih[o+1]
6          temp   ← a[o]
7          a[o]   ← a[o+1]
8          a[o+1] ← temp
```

Gambar 4.6. Implementasi Algoritma Bubble Sort

4.3.4. Menghitung Langkah

Dalam perhitungan langkah mebutuhkan kumpulan nilai koordinat axis teraktif. Telah terjadi suatu langkah apabila kurva nilai akselerasi telah melintasi *threshold*, dimana *threshold* didapat dengan rumus : (max+min)/2 dan kurva nilai akselerasi termasuk dalam kategori bergerak. Untuk mendapatkan kategori bergerak maka diperlukan uji coba sampel kategori bergerak.

4.3.4.1.Nilai Standar Kategori Bergerak

Nilai ini untuk menetukan pergerakan kaki sedang bergerak atau tidak. Dikategorikan bergerak ketika jarak antara nilai tertinggi untuk *positive slope* apabila nilai akselerasi sebelum lintasan *threshold* lebih besar dan nilai terendah untuk *negative slope* apabila nilai akselerasi sebelum *threshold* lebih rendah telah melebihi nilai standar kategori bergerak.

Nilai standar kategori bergerak ini adalah nilai rata-rata dari jarak antara nilai tertinggi atau terendah sebelum lintasan *threshold* terhadap nilai *threshold*. Nilai rata-rata jarak antar nilai tertinggi dan terendah dengan *threshold* berbeda-beda pada setiap sudutnya. Rata-rata nilai pada masing-masing sudut ditunjukan pada Tabel 4.2.

Tabel 4.2. Rata-rata jarak antar nilai tertinggi dan terendah dengan threshold pada setiap sdut

	Sudut		
	X	Y	Z
$\bar{x} = \sum \lvert\alpha – Threshold\rvert/n$	3,7	4,8	3,7

4.3.4.2.Implementasi Program

Setelah diketahui sudut teraktif, selanjutnya dilakukan implementasi algoritma pendeteksian langkah dan menghitung jumlah langkahnya. Untuk mendeteksi langkah diperlukan *threshold* dan nilai standar kategori bergerak. nilai *threshold* berubah secara dinamis, nilai berubah setiap 2 detik atau per proses ***loop*** arduino. Implementasi perhitungan nilai *threshold* ditunjukkan pada Gambar 4.7 dan implementasi algoritma deteksi langkah dan menghitung jumahnya ditunjukkan pada Gambar 4.8.

1. Kumpulan 18 nilai pada sudut akselerometer teraktif, dilakukan pengurutan menggunakan *bubble sort* untuk didapat koordinat tertinggi dan terendah.
2. Menghitung nilai *threshold* dengan menjumlahkan nilai koordinat tertinggi dan terendah lalu dibagi 2.
3. Proses penyeleksian nilai akselerasi koordinat axis teraktif dilakukan selama 18 kali.
4. Menyimpan nilai koordinat akselerasi secara sementara.
5. Apabila koordinat n lebih rendah dari *threshold* dan koordinat n-1 lebih besar dari *threshold* maka termasuk *negative slope*.
6. Melakukan pengurutan data nilai akselerasi yang telah disimpan secara sementara.
7. Menghapus isi data nilai sementara.
8. Menghitung jarak antara nilai tertinggi atau “bukit” dari nilai akselerasi yang telah diurut dengan *threshold*, bila jarak lebih atau sama dengan dari nilai standar kategori bergerak, maka termasuk ke dalam kategori langkah.
9. Apabila koordinat n lebih besar dari *threshold* dan koordinat n-1 lebih tinggi dari *threshold* maka termasuk *postive slope*.
10. Melakukan pengurutan data nilai akselerasi yang telah disimpan secara sementara.
11. Menghapus isi data nilai sementara.

12. Menghitung jarak antara nilai terendah atau "lembah" dari nilai akselerasi yang telah diurut dengan *threshold*, bila jarak lebih atau sama dengan dari standar yang ditentukan, maka termasuk ke dalam kategori langkah.

```
1  sort(sudut teraktif,18)
2  threshold ← (sudut[17] + sudut[0])/2
3  langkah ← countstep(sudut teraktif, threshold, 18,
   bergerak)
```

Gambar 4.7. Menghitung threshold

```
1   float function countstep
2     initialize (sudut_teraktif, threshold, size,
      bergerak)
3     i = 0
4     for j ← 1 to size - 1
5       temp[i] ← sudut_teraktif[j-1]
6       i++
7       if sudut_teraktif[j-1]<= batas AND
        sudut_teraktif[j]>=    batas
8          sort(temp,3)
9               if (|temp(min) − threshold |) > bergerak)
10                 i ← 0
11                 langkah++
12      else if sudut_teraktif[j-1]>= batas AND
        sudut_teraktif[j]<= batas
13         sort(temp,3)
14              if ((temp(max) - threshold) > bergerak)
15                 i ← 0
16                 langkah++
```

Gambar 4.8. Fungsi countstep

4.3.5. Menghitung *Stride*

Dalam menghitung *stride*, diperlukan inisialisasi tinggi badan(*height)* dan nilai langkah setiap 2 detik. output nilai *stride* didapat dari metode *fuzzy logic*.

Aturan *fuzzy* yang diimplementasikan adalah sebagai berikut:

1. IF langkah 2 detik = 0 or 1 or 2 THEN *stride* = Tinggi Badan /5
2. IF langkah 2 detik = 3 THEN *stride* = Tinggi Badan/4

3. IF langkah 2 detik = 4 THEN *stride* = Tinggi Badan/3
4. IF langkah 2 detik = 5 THEN *stride* = Tinggi Badan/2
5. IF langkah 2 detik = 6 THEN *stride* = Tinggi Badan /1.2
6. IF langkah 2 detik = 7 or 8 THEN *stride* = Tinggi Badan
7. IF langkah 2 detik = 9 THEN *stride* = Tinggi Badan * 1.2

4.3.6. Menghitung Kecepatan

Dalam menghitung kecepatan(*speed*), diperlukan nilai stride/2s dan nilai langkah/2s. Perhitungan *speed* dijelaskan sebagai berikut:

1. *Speed* = nilai stride/2s * nilai langkah/2s.

4.3.7. Menghitung Total Jarak

Dalam menghitung total jarak(*distance)*, pertama harus diketahui jarak/2s, untuk itu diperlukan nilai kecepatan dan untuk total jarak diperlukan nilai jarak/2s dengan total jarak sebelumnya dari EEPROM. Pada Gambar 4.9 ditunjukkan tahapan dalam menghitung nilai total jarak yang diimplementasikan sebagai berikut:

1. Jarak/2s = nilai kecepatan/2s dibagi 2 detik.
2. Mengambil data total jarak sebelumnya dari *high bytes* pada alamat 4 dan *low bytes* pada alamat 5 di dalam EEPROM untuk disatukan kedua data tersebut dengan fungsi *Word()*.
3. Data total jarak sebelumnya yang diperoleh dibagi 100.
4. Menghitung total jarak dengan menjumlahkan jarak/2s dan total jarak sebelumnya.

```
1  Jarak2s = speed2s * 2;
2  Jarak_EEPROM ← word(EEPROM.read(2),EEPROM.read(3))
```

```
3  Jarak_sebelumnya ← Jarak_EEPROM/100
4  Total Jarak ← Jarak Sebelumnnya + Jarak2s
```

Gambar 4.9. Menghitung Total Jarak

4.3.8. Menghitung Total Kalori yang Terbakar

Dalam menghitung Kalori(*calories)*, diperlukan nilai kecepatan setiap 2 detik, inisialisasi berat badan(*weight*) dan nilai 1800 adalah jumlah detik dalam satuan jam di interval 2 detik. Untuk total *calories* diperlukan nilai *calories* saat ini dengan total *calories* sebelumnya dari EEPROM. Pada Gambar 4.10 ditunjukkan tahapan dalam menghitung output nilai *calories* yang diimplementasikan sebagai berikut:

1. *calories* = 1.25 * nilai kecepatan(*speed)* setiap 2 detik * berat badan /1800. Dimana bisa dipersingkat dengan *calories* = nilai kecepatan(*speed)* setiap 2 detik * berat badan /400.
2. Mengambil data total *calories* sebelumnya dari *high bytes* pada alamat 4 dan *low bytes* pada alamat 5 di dalam EEPROM disatukan kedua data itu dengan fungsi *Word()*.
3. Data total *calories* yang diproleh dibagi 100.
4. Menghitung total *calories* dengan menjumlahkan *calories* dan total *calories* sebelumnya.

```
1  calories ← speed2s * weight/400
2  calories_EEPROM←word(EEPROM.read(4),EEPROM.read(5))
3  calories sebelumnya ← calories_EEPROM/100
4  Total calories ←  calories sebelumnya + calories
```

Gambar 4.10. Menghitung Total Kalori

4.3.9. Penyimpanan Data Di EEPROM

Untuk melakukan penyimpanan data ini, aplikasi membutuhkan nilai total langkah, total jarak yang ditempuh dan total kalori yang terbakar. Cara menyimpannya dengan mengkalikan 100 pada data dan membagi 2 dari *bytes* data menjadi *high bytes* dan *low bytes*. Setiap *bytes* disimpan pada

alamat yang berbeda. Pada Gambar 4.11 ditunjukkan tahapan dalam melakukan penyimpanan data di EEPROM dapat dijelaskan sebagai berikut:

1. Mengkalikan nilai total langkah, total jarak yang ditempuh dan total kalori yang terbakar dengan 100.
2. Membagi *bytes* data total lagkah, total jarak yang ditempuh dan total kalori yang terbakar, menjadi *high bytes* dan *low bytes.*
3. Menyimpan masing-masing *bytes* pada alamat yang berbeda.

```
1   Total langkah ← Total langkah * 100
2   byte highlangkah ← highByte(total langkah)
3   byte lowlangkah ← lowByte(total langkah)
4   Total Jarak ← Total Jarak * 100
5   byte highdistance ← highByte(Total Jarak)
6   byte lowdistance ← lowByte(Total Jarak)
7   Total calories ← Total calories * 100
8   byte highcalories ← highByte(total calories)
9   byte lowcalories ← lowByte(total calories)
10  EEPROM[0] ← highlangkah
11  EEPROM[1] ← lowlangkah
12  EEPROM[2] ← highdistance
13  EEPROM[3] ← lowdistance
14  EEPROM[4] ← highcalories
15  EEPROM[5] ← lowcalories
```

Gambar 4.11. Penyimpanan data total langkah, total jarak dan total kalori

4.3.10. *Reset* Isi Data Pada EEPROM menjadi 0

Pada fungsi *reset* isi data pada EEPROM menjadi 0, data yang di-*reset* adalah data total langkah, total jarak yang ditempuh dan total kalori yang terbakar. Cara me-*reset* adalah dengan mengisi data 0 pada alamat masing-masing. Pada Gambar 4.12 ditunjukkan tahapan dalam melakukan *reset* data pada EEPROM dapat dijelaskan sebagai berikut:

1. Melakukan pengecekan jika *serial* tersedia.
2. Membaca nilai inputan dari serial.
3. Jika nilai inputan “1” maka dijalankan fungsi *reset* .

4. Menghitung jumlah alamat EEPROM.
5. Melakukan fungsi *write* pada setiap alamat EEPROM dengan nilai 0.

```
1  if Serial.available > 0
2     inbyte = Serial.read
3     if inbyte ← 1
4        for c ← 0 to EEPROM.length()
5            EEPROM.write(c,0)
```

Gambar 4.12. Melakukan Reset Isi Data Pada EEPROM Menjadi 0

4.4. Implementasi Pada Aplikasi Android

Aplikasi android ini digunakan oleh *user*. Implementasi yang dilakukan pada aplikasi android ini berguna untuk melakukan pemantauan nilai total lngkah, total jarak dan total kalori, selain itu untuk berfungsi melakukan *reset* data.

4.4.1. Implementasi Pengaktifan *Bluetooth*

Bagian ini menjelaskan tentang pengecekan *bluetooth* bagaimana sebuah *bluetooth* telah tersedia di handphone dan mengaktifkan *bluetooth* yang ditunjukkan pada gambar 4.13 dengan penjelasan sebagai berikut:

1. Melakukan pengecekan jika handphone telah memiliki perangkat *bluetooth*.
2. Melakukan pengecekan jika *bluetooth* telah diaktifkan
3. Jika *bluetooth* belum diaktifkan maka akan memunculkan sebuah *decision popup* untuk mengaktifkan *bluetooth*.

```
1  void function CheckBluetoothState
2    set getDefaultAdapter to mBTAdapter
3    If mBTAdapter == null
4       return false
5    else
6      if bluetooth == on
7        return true
8      else
```

```
9           enableBT ← intent
            BluetoothAdapter.ACTION_REQUEST_ENABLE
10          startActivityForResult(enableBtIntent, 1)
```

Gambar 4.13. Fungsi CheckBluetoothState

4.4.2. Implementasi Melakukan *Pair* Dengan *Bluetooth* Mikrokontroler

Bagian ini menjelaskan tentang bagaimana sebuah *bluetooth* handphone melakukan pair dengan *bluetooth* mikrokontroler. Proses ini akan dibagi dengan 2 proses yaitu mengumpulkan *list bluetooth device* yang ditunjukkan pada Gambar 4.14 dan memilih *device bluetooth* dengan menggunakan fitur *onClick* yang ditunjukkan pada Gambar 4.15 dengan penjelasan sebagai berikut:

1. Melakukan pengumpulan list *bluetooth device* yang telah di *pair* menggunakan *bluetooth* handphone.
2. Melakukan pemilihan *device* dengan menggunakan fungsi *onclick*.
3. Membuat fungsi *intent* untuk Perpindahan halaman dari halaman *list bluetooth devices* ke monitoring kalori.
4. Menjalankan fungsi *intent*.

```
1   pairedListView.setAdapter(mPairedDevicesArrayAdapter)
2   pairedListView.setOnItemClickListener(mDeviceClick
    Listener)
3   set getBondedDevices to pairedDevices
4     If pairedDevices > 0
5        for BluetoothDevice to pairedDevices.size
6          mPairedDevicesArrayAdapter.add(name,address)
7     else
8        return false
```

Gambar 4.14. Melakukan pair dengan blluetooth mikrokontroler

```
1   private AdapterView mDeviceClickListener
2     void onItemClick(AdapterView)
3      intent i ← new Intent(DeviceListActivity.this,
      MainActivity.class)
4      startActivity(i)
```

Gambar 4.15. Fungsi onItemClick

4.4.3. Implementasi Monitoring Data Dari Mikrokontroler

Setelah handphone melakukan pair dengan *bluetooth* mikrokontroler, akan langsung menerima data dari mikrokontroler arduino. Pada Gambar 4.16 merupakan cara aplikasi android menerima data dari mikrokontroler dengan penjelasan sebagai berikut:

1. Melakukan inisialisasi stringbuilder *recDataString*
2. Melakukan inisialisasi string *readMessage*.
3. String *readMessage* menerima pesan dari mikrokontroler.
4. Pesan dari string *readmessage* di simpan di stringbuilder *recDataString.*
5. Mengecek huruf terakhir pada pesan apakah bertanda "#" yang menunjukan akhir dari setiap baris pesan tersebut.
6. Menghapus tanda "#".
7. Melakukan pemisahan pesan berdasarkan tanda ",". Dimana setiap potongan pesan itu disimpan ke dalam array dengan alamat yang berbeda.
8. Mendapatkan data total langkah, total jarak dan total kalori dari potongan array yang dirubah bentuk datanya dari string ke *float.*
9. Menghapus data dari *recDataString* untuk pengisian recDataString selanjutnya.

```
    private StringBuilder recDataString ← new
    StringBuilder();
1   void handlemessage
2   if msg.what ← handlerState
3   string readMessage
4   readMessage ←  msg.obj
5   append readMessage ← recDataString
6   int endOfLineIndex to recDataString index of "#"
7     if endOfLineIndex > 0
8       dataInPrint ←
        recDataString.substring(0,endOfLineIndex)
9       int dataLength ← dataInPrint.length()
10      delete char recDataString index of "#"
```

```
11      myArray ← recDataString.toString() split(",")
12      step ← Float.parseFloat(myArray[0])
13      distance ← Float.parseFloat(myArray[1])
14      calories ← Float.parseFloat(myArray[2])
15  delete recDataString from char 0 to
    recDataString.length()
```

Gambar 4.16. Algoritma menerima pesan dari mikrokontroler

4.4.4. Implementasi Perintah *Reset* Data Menjadi 0

Data total langkah, total jarak yang ditempuh dan total kalori yang terbakar pada EEPROM dapat *reset* menjadi 0 dengan perintah dari aplikasi android menggunakan fitur *onClick button* pada aplikasi android yang ditunjukkan pada Gambar 4.17 dengan penjelasan sebagai berikut:

1. Melakukan pemilihan *button* yang memiliki fitur *onclick.*
2. Mengirim pesan berupa angka “1” ke mikrokontroler.

```
1  btnOn.setOnClickListener(new OnClickListener()
2   void onClick
3    Send "1"
```

Gambar 4.17. Fungsi onClick

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dibahas mengenai uji coba dari segi fungsionalitas dan performa dari aplikasi. Uji coba fungsionalitas dan performa akan dibagi ke dalam beberapa skenario uji coba.

5.1. Lingkungan Uji Coba

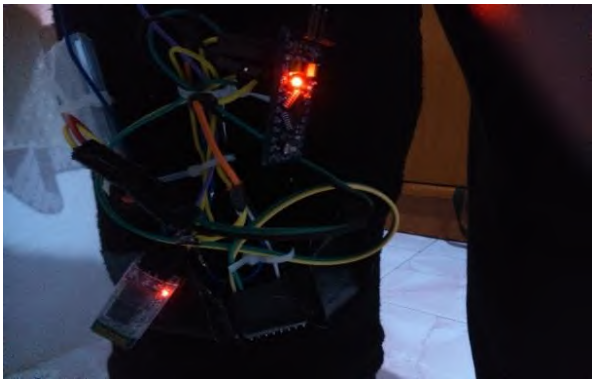
Pada subbab ini, dijelaskan mengenai gambaran lingkungan yang digunakan sebagai uji coba program yang dibangun. Uji coba dilakukan di lorong gedung A Jurusan Teknik Informatika ITS yang memiliki medan yang datar.

Perangkat mikrokontroler direkatkan pada sebuah knee pad yang dipasang di salah satu bgaian lutut, penggunaan bisa di lutut pada kaki kanan maupun kaki kiri. Pemakaian di lutut karena tingkat sensitivitas pada nilai akselerasi yang tinggi terhadap pergerakan kaki.

Pengiriman data dari perangkat mikrokontroler ke aplikasi android di perangkat bergerak (*mobile*) berbasis aplikasi android dilakukan secara *real time* menggunakan koneksi *wireless* yaitu *bluetooth*. Lingkungan uji coba memiliki spesifikasi sebagai berikut:

1. Perangkat mikrokontroler (mikrokontroler Arduino, sensor Akselerometer, *bluetooth module*, baterai *li-po*, satu *set* kabel),
2. Knee pad berukuran sesuai lutut pengguna,
3. Hand Counter,
4. Alat meteran putar gulung,
5. Perangkat bergerak (*mobile)* berbasis android yaitu, Handphone Sony Xperia L

Spesifikasi yang telah dijabarkan di atas merupakan spesifikasi lingkungan uji coba yang digunakan selama pengerjaan uji coba. Gambaran lingkungan uji coba ditunjukkan seperti pada Gambar 5.1.



Gambar 5.1. Lingkungan Uji Coba

5.2. Uji Coba Fungsionalitas

Uji coba fungsionalitas merupakan sebuah pengujian terhadap jalannya fungsi-fungsi utama yang ada pada aplikasi. Pada aplikasi ini, sistem terbagi menjadi dua bagian, yaitu perangkat mikrokontroler Arduino dan aplikasi android. Uji coba fungsionalitas dilakukan pada kedua bagian sistem tersebut. Fungsionalitas utama dari aplikasi terdapat pada mikrokontroler Arduino dan aplikasi Android. Program ditanamkan dan dijalankan pada mikrokontroler dan di-*monitor* pada aplikasi Android. Keberhasilan pada uji coba ini akan menunjang kualitas dari uji coba yang selanjutnya yaitu uji coba performa atau tingkat akurasi sistem.

5.2.1. Uji Coba Monitoring Data Mikrokontroler

Uji coba ini dilakukan untuk menguji fitur aplikasi monitoring. Dari fitur ini, dapat dilihat hasil data total langkah, total jarak. yang ditempuh dan total kalori yang terbakar dari pengolahan data mikrokontroler. Data tersebut ditampilkan pada layar Android. Pada Gambar 5.2. Ditunukkan Data yang diterima

Gambar 5.2. Tampilan Data Total Langkah, Total Jarak Dan Total Kalori Yang Terbakar Pada Aplikasi Android

5.2.2. Uji Coba Melakukan *Reset* Data Menjadi 0

Uji coba melakukan *reset* data menjadi 0 dilakukan untuk mengubah isi nilai data total langkah, total jarak yang ditempuh dan total kalori yang terbakar menjadi 0 pada EEPROM. Pengguna dapat melakukan *reset* data menjadi 0 dengan cara menekan tombol *clear data* pada perangkat bergerak yang terdapat aplikasi di android. Setelah pengguna menekan tombol *clear data* maka program akan memberikan perintah ke perangkat mikrokontroler untuk mengisi angka 0 pada alamat EEPROM untuk total langkah, total jarak yang ditempuh, total kalori yang terbakar dan menampikan sebuah pesan notifikasi "*clear data*" pada aplikasi android. Seperti ditunjukkan pada Gambar 5.3. waktu proses untuk melakukan *reset* data memakan waktu yang cukup besar.

Gambar 5.3. Tampilan Reset Data Menjadi 0, Muncul Pesan Notifikasi "clear data"

Pada gambar 5.3 nilai total kalori tidak berubah, tapi ini sudah mengalami perubahan menjadi 0 karena jeda waktu antara *screenshot* aplikasi dengan proses setiap ***loop*** arduino yang cepat mengakibatkan nilai telah berubah.

5.3. Uji Coba Performa

Pada bagian ini akan dilakukan uji coba performa atau tingkat akurasi algoritma ketika diimplementasikan pada keadaan *real-time*. Uji coba performa akan menggunakan *knee pad* dan uji coba akan dilakukan di lingkungan gedung Teknik Informatika ITS pada lorong gedung A lantai 3. Terdapat 4 skenario uji coba yang dilakukan, yaitu uji coba deteksi langkah dan jarak, uji coba kalori, *storage consumption* EEPROM dan uji coba lama waktu pada setiap proses. Uji coba tersebut akan dijelaskan pada masing-masing subbab.

5.3.1. Uji Coba Deteksi Langkah dan jarak

Uji coba yang paling utama adalah uji coba akurasi deteksi langkah dan jarak.

5.3.1.1. Skenario dan Hasil Uji Coba Deteksi Langkah dan Jarak

Skenario uji coba dilakukan ke dalam 4 kategori jarak, yaitu 5 meter, 10 meter, 25 meter, 50 meter dan 100 meter dengan kondisi berjalan pada medan yang datar.

Untuk menilai akurasi langkah dilakukan dengan membandingkan hasil total langkah dari mikrokontroler yang dapat dilihat dari aplikasi monitoring dengan hasil dari alat hand counter yang dipegang dengan tangan. Untuk penggunaan alat counter dilakukan dengan cara jempol akan menekan alat counter untuk setiap langkahnya.

Untuk menilai akurasi jarak dilakukan dengan membandingkan hasil total langkah dari mikrokontroler yang dapat dilihat dari aplikasi monitoring dengan total jarak yang ditempuh di lorong gedung A yang telah diukur secara konvensional menggunakan alat meteran putar gulung.

Data perbandingan jumlah langkah dan jarak pada mikrokonroler dengan kondisi nyata dapat dilihat pada Tabel 5.1, Tabel 5.2, Tabel 5.3, Tabel 5.4 dan tabel 5.5.

Tabel 5.1. Data jumlah langkah dan jarak dari mikrokontroler dibanding jumlah langkah yang sebenarnya pada jarak 5 meter

No	Langkah		Jarak (meter)	
	Mikrokontroler	Kondisi Nyata	Mikrokontroler	Kondisi Nyata
1	12	10	5,23	5
2	10	10	3,65	5
3	9	9	3,31	5

No	Langkah		Jarak (meter)	
	Mikrokontroler	Kondisi Nyata	Mikrokontroler	Kondisi Nyata
4	11	10	4,64	5
5	11	10	4,23	5
6	10	10	3,65	5

Tabel 5.2. Data jumlah langkah dan jarak dari mikrokontroler dibanding jumlah langkah yang sebenarnya pada jarak 10 meter

No	Langkah		Jarak (meter)	
	Mikrokontroler	Kondisi Nyata	Mikrokontroler	Kondisi Nyata
1	19	18	8,76	10
2	20	18	9,84	10
3	20	18	8,44	10
4	18	18	7,77	10
5	17	18	7,58	10
6	18	18	9,17	10

Tabel 5.3. Data jumlah langkah dan jarak dari mikrokontroler dibanding jumlah langkah yang sebenarnya pada jarak 25 meter

No	Langkah		Jarak (meter)	
	Mikrokontroler	Kondisi Nyata	Mikrokontroler	Kondisi Nyata
1	45	46	20,25	25
2	48	43	21,27	25
3	45	42	19,73	25
4	43	41	20,37	25
5	44	41	24,76	25
6	41	41	17,48	25

Tabel 5.4. Data jumlah langkah dan jarak dari mikrokontroler dibanding jumlah langkah yang sebenarnya pada jarak 50 meter

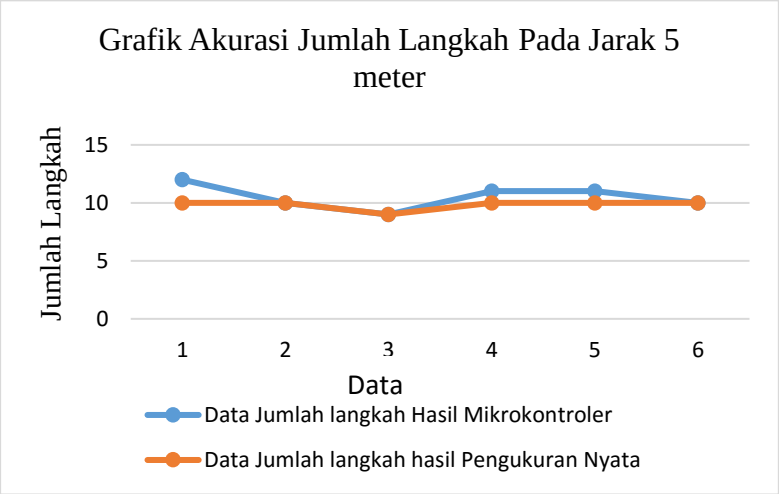
No	Langkah		Jarak (meter)	
	Mikrokontroler	Kondisi Nyata	Mikrokontroler	Kondisi Nyata
1	93	83	48,76	50
2	93	85	50,18	50
3	92	83	48,13	50
4	86	81	48,45	50
5	94	78	53,26	50
6	88	79	46,22	50

Tabel 5.5. Data jumlah langkah dan jarak dari mikrokontroler dibanding jumlah langkah yang sebenarnya pada jarak 100 meter

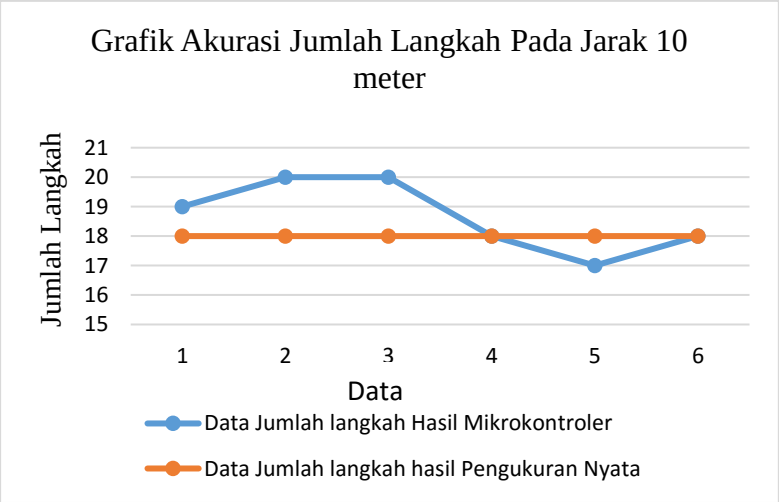
No	Langkah		Jarak (meter)	
	Mikrokontroler	Kondisi Nyata	Mikrokontroler	Kondisi Nyata
1	174	156	114	100
2	175	151	102,8	100
3	169	157	84,66	100
4	152	150	74,55	100
5	175	159	88,45	100
6	168	153	99,17	100

Setelah melakukan percobaan, maka harus dilakukan evaluasi antara data langkah nyata dan data langkah yang didapatkan dari hasil mikrokontroler. Selain itu juga perlu dilakukan evaluasi antara data jarak nyata dan data jarak hasil mikrokontroler. Grafik hasil akurasi jumlah langkah dari mikrokontroler pada percobaan 1 dapat dilihat pada gambar 5.4, gambar 5.5, gambar 5.6, gambar 5.7 dan gambar 5.8. Grafik hasil

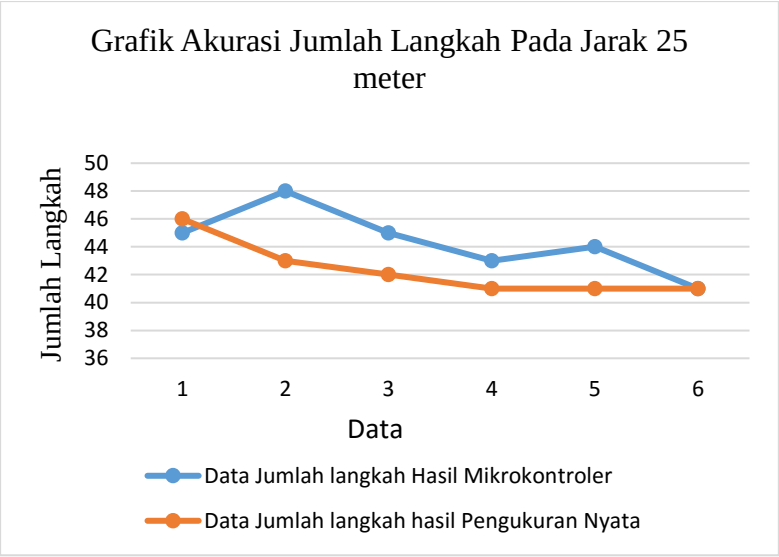
akurasi jarak dari mikrokontroler dapat ilihat pada gambar 5.9, gambar 5.10, gambar 5.11, gambar 5.12 dan gambar 5.13.



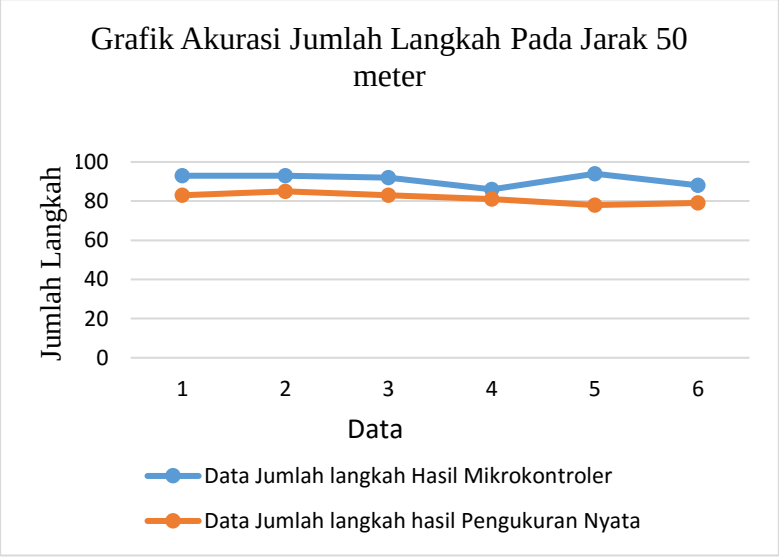
Gambar 5.4. Grafik Akurasi Jumlah Langkah Pada Jarak 5 meter



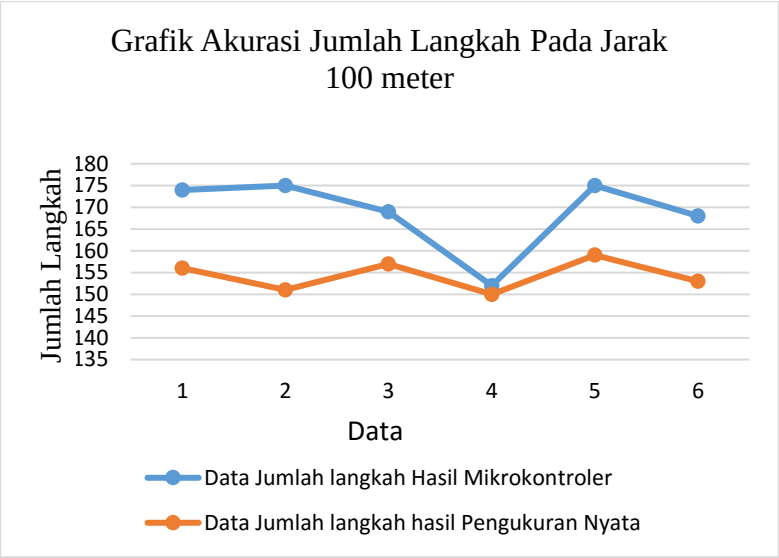
Gambar 5.5. Grafik Akurasi Langkah Pada Jarak 10 meter



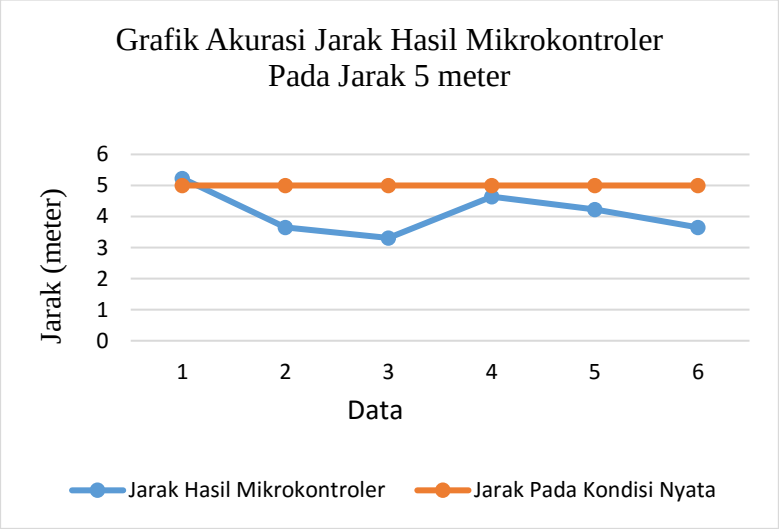
Gambar 5.6. Grafik Akurasi Jumlah Langkah Pada Jarak 25 meter



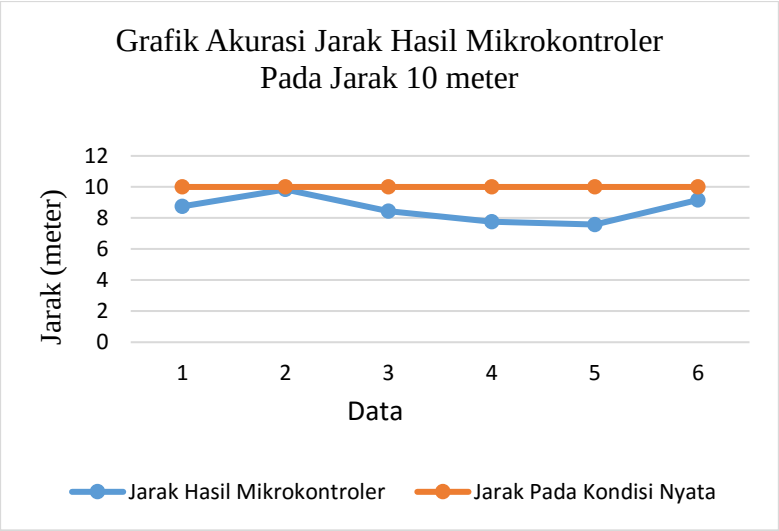
Gambar 5.7. Grafik Akurasi Jumlah Langkah Pada Jarak 50 meter



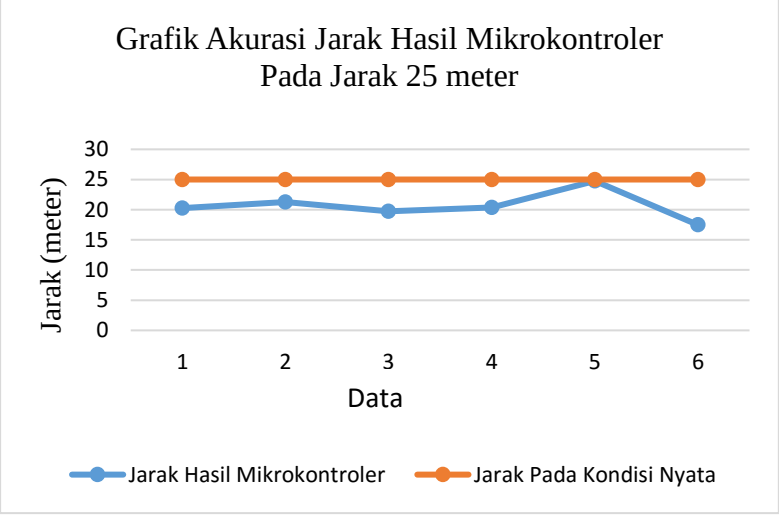
Gambar 5.8. Grafik Akurasi Jumlah Langkah Pada Jarak 100 meter



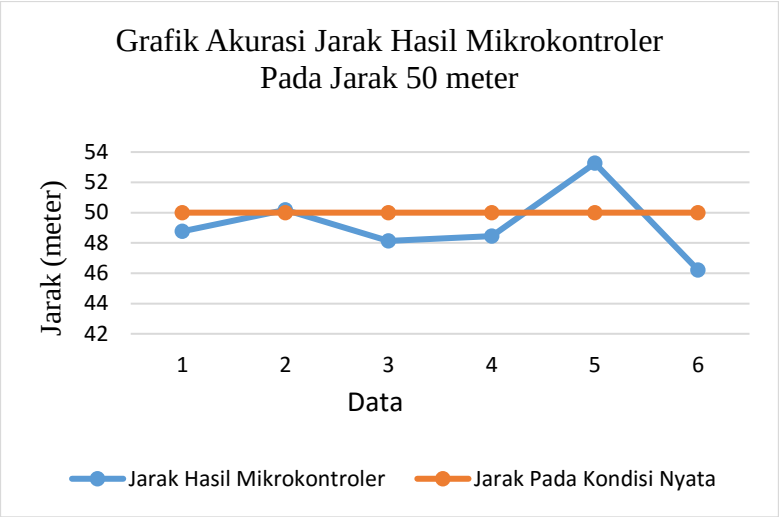
Gambar 5.9. Grafik Akurasi Jarak Hasil Mikrokontroler Pada Jarak 5 meter



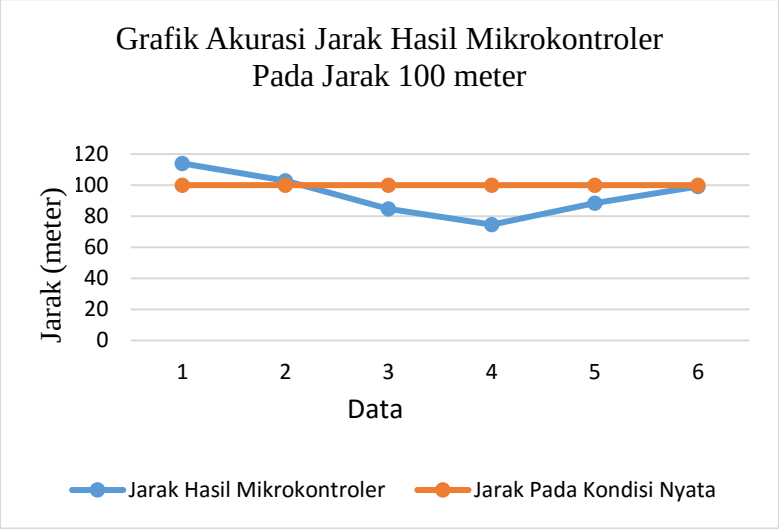
Gambar 5.10. Grafik Akurasi Jarak Hasil Mikrokontroler Pada Jarak 10 meter



Gambar 5.11. Grafik Akurasi Jarak Hasil Mikrokontroler Pada Jarak 25 meter



Gambar 5.12. Grafik Akurasi Jarak Hasil Mikrokontroler Pada Jarak 50 meter



Gambar 5.13. Grafik Akurasi Jarak Hasil Mikrokontroler Pada Jarak 100 meter

5.3.1.2. Analisa Hasil Uji Coba Deteksi Langkah dan Jarak

Berdasarkan skenario uji coba deteksi langkah dan jarak, didapatkan tingkat akurasi dari langkah dan jarak. Perbandingan tingkat akurasi pada percobaan deteksi langkah dapat dilihat pada Tabel 5.5 dan tingkat akurasi jarak pada Tabel 5.6.

Tabel 5.6. Perbandingan Tingkat Akurasi Langkah

No	Tingkat Akurasi Langkah Per Uji Coba untuk Setiap Jarak (meter)				
	5	10	25	50	100
1	80%	94,44%	97,83%	87,95%	88,46%
2	100%	88,88%	88,37%	90,59%	84,11%
3	100%	88,88%	92,86%	89,16%	92,36%
4	90%	100 %	95,12%	93,82%	98,67%
5	90%	94,44%	92,68%	79,49%	89,94%
6	100%	100%	100%	88,6%	90,2%
Rata-rata Akurasi/Jarak	93,33%	94,44%	94,48%	88,27%	90,62%
Rata-rata Akurasi	92,22%				

Tabel 5.7. Perbandingan Tingkat Akurasi Jarak

No	Tingkat Akurasi Jarak Per Uji Coba (meter)				
	5	10	25	50	100
1	95,4%	87,6%	81%	97,52%	86%
2	73%	98,4%	85,08%	99,64%	97,2%
3	66,2%	84,4%	78,92%	96,26%	84,66%
4	92,8%	77,7%	81,48%	96,9%	74,55%
5	84,6%	75,8%	99,04%	93,48%	88,45%
6	73%	91,7%	69,92%	92,44%	99,17%

No	Tingkat Akurasi Jarak Per Uji Coba (meter)				
	5	10	25	50	100
Rata-rata Akurasi/Jarak	80,33%	85,93%	82,57%	96,04%	88,83%
Rata-rata Akurasi	86,64%				

Tabel 5.8. Tingkat Kecenderungan Hasil langkah Mikrokontroler dengan Kondisi Nyata

Kondisi Perbandingan Hasil Langkah	Tingkat Kecenderungan Berdasarkan Jarak (meter)					Rata-rata
	5	10	25	50	100	
Mikrokontroler < Kondisi Nyata	-	16,67%	16,67%	-	-	6,67%
Mikrokontroler = Kondisi Nyata	50%	33,33%	16,67%	-	-	20%
Mikrokontroler > Kondisi Nyata	50%	50 %	66,67%	100%	100%	73,33%

Tabel 5.9. Tingkat Kecenderungan Hasil Jarak Mikrokontroler dengan Kondisi Nyata

Kondisi Perbandingan Hasil Jarak	Tingkat Kecenderungan Berdasarkan Jarak (meter)					Rata-rata
	5	10	25	50	100	
Mikrokontroler < Kondisi Nyata	83,33%	66,67%	100%	66,67%	66,67%	76,67%
Mikrokontroler = Kondisi Nyata	-	-	-	-	-	-
Mikrokontroler > Kondisi Nyata	16,67%	33,33%	-	33,33%	33,33%	33,33%

Pada skenario uji coba deteksi langkah dan jarak, dapat dilihat bahwa akurasi langkah lebih akurat dibanding akurasi jarak, dengan rata-rata akurasi langkah adalah 92,22% sedangkan akurasi jarak adalah 86,64%. Dengan pola grafik akurasi dari langkah dan jarak cenderung tidak menentu meskipun sudah berjalan semakin jauh.

Tingkat kecenderungan hasil langkah dari mikrokontroler dengan kondisi nyata, seperti yang tertera pada Tabel 5.8 adalah 6,67% untuk hasil mikrokontroler lebih kecil dari kondisi nyata, 20% untuk hasil mikrokontroler sama dengan kondisi nyata dan 73,33% untuk hasil mikrokontroler lebih besar dari kondisi nyata. Sedangkan untuk tingkat kecenderungan hasil jarak dari mikrokontroler dengan kondisi nyata, seperti yang tertera pada Tabel 5.9 adalah 76,67% untuk hasil mikrokontroler lebih kecil dari kondisi nyata, tidak ada data dari hasil mikrokontroler yang sama dengan kondisi nyata dan 23,33% untuk hasil mikrokontroler lebih besar dari kondisi nyata.

Nilai mayoritas rata-rata perbandingan tingkat kecenderungan hasil langkah dan jarak saling berbanding terbalik, dimana pada hasil langkah, rata-rata hasil dari mikrokontroler lebih besar dari kondisi nyata sedangkan pada hasil jarak, justru rata-rata hasil dari mikrokontroler lebih kecil dari kondisi nyata.

5.3.2. Uji Coba Kalori

Uji coba ini dilakukan untuk memperoleh nilai dari jumlah kalori yang terbakar dari pengguna perangkat mikrokontroler saat berjalan pada medan datar.

5.3.2.1. Skenario dan Hasil Uji Coba Kalori

Skenario uji coba dilakukan pada jarak 100 meter. Data perbandingan kalori hasil pada mikrokontroler dengan hasil aplikasi android yang dapat menghitung pembakaran kalori, pada

uji coba ini menggunakan aplikasi pembanding yang memiliki nilai *ratings* yang tinggi dengan jumlah pengunduh yang besar yang dapat dilihat pada aplikasi google play store, yaitu endomondo dan my tracks. Kedua aplikasi pembanding ini melakukan perhitungan kalori dari hasil pengolahan data jarak yang ditempuh berdasarkan sensor GPS(*Global Positioning System*).

Uji coba dilakukan dengan berjalan kaki di lorong gedung A Teknik Informatka ITS dengan panjang 50 meter secara 2 putaran yang memiliki medan yang datar. Hasil nilai kalori dari perangkat mikrokontroler, aplikasi my tracks dan aplikasi endomondo dapat dilihat pada Tabel 5.10. Hasil dari mikrokontroler langsung dilihat pada aplikasi monitoring ketika berjalan sudah mencapai 100 meter

Tabel 5.10. Hasil Uji Coba Mikrokontroler dengan Aplikasi My Tracks Dan Endomondo Pada Jarak 100 meter

		Uji Coba					
		1	**2**	**3**	**4**	**5**	**6**
Kalori (Kcal)	**Mikrokontroler**	10,03	8,87	7,31	6,69	7,65	8,4
	My Tracks	7	7	8	8	7	8
	Endomondo	8	9	9	8	9	9

5.3.2.2. Analisa Hasil Uji Coba Kalori

Berdasarkan hasil dari skenario uji coba kalori pada Tabel 5.10, didapatkan tingkat presisi dari nilai kalori perangkat mikrokontroler terhadap rata-rata nilai kalori dari dua aplikasi pembanding pada setiap uji coba. Perbandingan tingkat presisi perangkat mikrokontroler terhadap kedua aplikasi pembanding pada percobaan kalori dapat dilihat pada Tabel 5.11, dan tingkat kecenderungan kalori hasil mikrokontroler terhadap kedua aplikasi pembanding yaitu My Tracks dan Endomondo dapat dilihat pada Tabel 5.12.

Tabel 5.11. Tingkat Presisi Kalori Mikrokontroler Terhadap Aplikasi Pembanding

No	Mikrokontroler	Rata-rata Aplikasi My Tracks dan Endomondo	Tingkat Presisi Mikrokontroler Terhadap Aplikasi
1	10,03	7,5	66,27%
2	8,87	8	89,13%
3	7,31	8,5	86%
4	6,69	8	83,63%
5	7,65	8	95,63%
6	8,4	8,5	98,82%
Rata-rata Presisi Kalori			86,58%

Tabel 5.12. Tingkat Kecenderungan Hasil Kalori Mikrokontroler dengan Aplikasi My Tracks Dan Endomondo

Kondisi Perbandingan Kalori	Tingkat Kecenderungan
Mikrokontroler < Aplikasi	33,33%
Mikrokontroler = Aplikasi	-
Mikrokontroler > Aplikasi	66,67%

Tingkat kecenderungan hasil presisi kalori dari mikrokontroler terhadap aplikasi my tracks dan endomondo, seperti yang tertera pada Tabel 5.12 adalah 33,33% untuk hasil mikrokontroler lebih kecil dari aplikasi My tracks dan Endomondo, tidak ada nilai yang sama antara mikrokontroler dengan kedua aplikasi tersebut dan 66,67% untuk hasil mikrokontroler lebih besar dari aplikasi My tracks dan Endomondo.

5.3.3. *Storage Consumption* EEPROM

Uji coba performa ini menjabarkan sistem penyimpanan data pada EEPROM. Seperti diketahui EEPROM menyimpan data

setiap kurang lebih 2 detik dengan data yang disimpan adalah total langkah, total jarak dan total kalori. dimana masing-masing nilai data ini bisa dapat lebih dari 8 bit atau nilainya diluar angka 0 sampai 255, maka dari itu setiap nilai dibagi menjadi menjadi 2 bytes, yaitu *high byte* dan *low byte*. *Storage consumption* bergantung pada ata tersebut masuk ke dalam kategori bergerak atau tidak. Untuk kategori bergerak menyimpan itu semua membutuhkan 6 alamat atau 6 *bytes*. Untuk proses data kategori tidak bergerak akan menghabiskan 2 alamat atau 2 *bytes*, Penyimpanan itu hanya untuk nilai total kalori. Contoh *storage consumption* pada setiap proses tertera pada Tabel 5.13.

Tabel 5.13. Storage Consumption pada Setiap Proses

No	Total Langkah	Total Jarak (meter)	Total Kalori	*Storage Consumption (bytes)*
1	0	0	0,4	2
2	0	0	0,7	2
3	2	0,68	0,12	6
4	4	1,36	0,17	6
5	6	2,04	0,22	6
6	7	2,38	0,24	6
7	10	3,64	0,37	6

5.3.4. Uji Coba Lama Waktu pada Setiap Proses

Uji coba performa ini menampilkan lama waktu setiap proses ***loop*** arduino, dimana setiap proses akan menghabiskan kurang lebih sekitar 2 detik.

5.3.4.1. Hasil Uji Coba Lama Waktu pada Setiap Proses

Hasil uji coba lama waktu pada setiap proses dapat dilihat pada Tabel 5.14. waktu proses ditampilkan pada satuan *miliseconds*(ms).

Tabel 5.14. Hasil Uji Coba Lama Waktu pada Setiap Proses

No	Hasil dari Mikrokontroler per proses			
	Langkah	Jarak (meter)	Kalori	Waktu proses (ms)
1	0	0	0,03	1956
2	2	0,68	0,05	1964
3	0	0	0,03	1954
4	1	0,34	0,02	1966
5	2	0,68	0,05	1962
6	0	0	0,03	1956
7	0	0	0,03	1954
8	2	0,68	0,05	1964
9	0	0	0,03	1956
10	2	0,68	0,05	1964
11	0	0	0,03	1952

5.3.4.2. Analisa Hasil Uji Coba Lama Waktu pada Setiap Proses

Berdasarkan hasil dari mikrokontroler untuk lama waktu pada setiap proses, didapatkan pengkategorian pada setiap proses, terdapat 2 kategori, yaitu proses bergerak dan proses tidak bergerak. Setiap proses akan dikategorikan pada salah satu kateori tersebut, Proses bergerak adalah proses yang termasuk dalam kategori nilai bergerak, di Tabel 5.14 ditunjukan dengan adanya nilai lebih dari 0 pada bagian langkah dan jarak, maka itu dari Tabel 5.14 , setiap proses diklasfikasikan termasuk bergerak atau tidak bergerak, pengkategorian ditunjukkan pada Tabel 5.15 dan rata-rata waktu masing-masing dari 2 kategori tersebut ditunjukkan pada Tabel 5.16.

Tabel 5.15. Pengambilan Keputusan Setiap Proses

No	Kondisi
1	Tidak Bergerak
2	Bergerak
3	Tidak Bergerak
4	Bergerak
5	Bergerak
6	Tidak Bergerak
7	Tidak Bergerak
8	Bergerak
9	Tidak Bergerak
10	Bergerak
11	Tidak Bergerak

Tabel 5.16. Rata-rata Waktu Setiap Proses Pada Kondisi Tidak Bergerak dan Kondisi Bergerak

Kondisi	Rata-rata Waktu Proses (*miliseconds*)
Proses Kondisi Tidak Bergerak	1954,625
Proses Kondisi Bergerak	1964

Dari Tabel 5.15. diapatkan terdapat 5 proses bergerak dan 6 proses tidak bergerak. Hasil Rata-rata Waktu Setiap Proses yang ditunjukan Tabel 5.16. diketahui setiap proses pada kondisi tidak bergerak menghabiskan waktu 1954,625 *miliseconds* dan pada kondisi bergerak menghabiskan waktu 1964 *miliseconds*, bisa ditarik kesimpulan bahwa lama waktu proses perhitungan pada kondisi bergerak lebih lama dibandingkan proses tidak bergerak.

LAMPIRAN

Tabel A. 1 Data *training* sensor *accelerometer* pengguna berjalan

No.	X	Th_x	Y	Th_y	Z	Th_z
1.	-21,6	-18,4	4,28	-1,35	43,9	47,8
2.	-17,6	-18,4	4,79	-1,35	45,3	47,8
3.	-17	-18,4	-3,8	-1,35	48,6	47,8
4.	-18,5	-18,4	-1,92	-1,35	50,1	47,8
5.	-19	-18,4	-0,35	-1,35	50,7	47,8
6.	-19,5	-18,4	-0,24	-1,35	50,1	47,8
7.	-22,2	-18,4	-2,82	-1,35	50,6	47,8
8.	-18,8	-18,4	4,24	-1,35	44,6	47,8
9.	-15	-18,4	1,37	-1,35	46,7	47,8
10.	-15,7	-18,4	-1,92	-1,35	50,2	47,8
11.	-14,7	-18,4	-7,49	-1,35	46,1	47,8
12.	-18,9	-18,4	-0,12	-1,35	51,6	47,8
13.	-19,9	-18,4	-0,04	-1,35	48,9	47,8
14.	-24,5	-19,4	0,39	-0,67	51,2	47,3
15.	-13,7	-19,4	3,45	-0,67	44,4	47,3
16.	-19,2	-19,4	-4,9	-0,67	46,8	47,3
17.	-25,6	-19,4	2,82	-0,67	52,3	47,3
18.	-18,2	-19,4	3,45	-0,67	49,6	47,3
19.	-18,3	-19,4	1,69	-0,67	50,1	47,3
20.	-20,4	-19,4	0,63	-0,67	52,8	47,3
21.	-18,1	-19,4	4,94	-0,67	41,7	47,3
22.	-13,3	-19,4	1,92	-0,67	44,4	47,3
23.	-15,9	-19,4	-8,98	-0,67	47,2	47,3
24.	-18,6	-19,4	7,65	-0,67	46,5	47,3
25.	-19,3	-19,4	-0,47	-0,67	52,3	47,3

No.	X	Th_x	Y	Th_y	Z	Th_z
26.	-19,9	-19,4	0,08	-0,67	49,2	47,3
27.	-26,1	-18,8	-1,88	1,65	50,1	47,9
28.	-12,6	-18,8	5,26	1,65	43,3	47,9
29.	-11,5	-18,8	-0,39	1,65	47,8	47,9
30.	-13,1	-18,8	4,59	1,65	50,1	47,9
31.	-19,7	-18,8	8,24	1,65	50,2	47,9
32.	-17,8	-18,8	0,27	1,65	48,6	47,9
33.	-17	-18,8	0,31	1,65	47,2	47,9
34.	-19,3	-18,8	-4,94	1,65	52,5	47,9
35.	-17,7	-18,8	3,18	1,65	50	47,9
36.	-19,3	-18,8	0,2	1,65	50,6	47,9
37.	-19	-18,8	0,51	1,65	50,8	47,9
38.	-20,6	-18,8	-0,27	1,65	49,9	47,9
39.	-17,2	-18,8	0,75	1,65	48,2	47,9
40.	-19	-18,8	-3,02	1,65	47,5	47,9
41.	-17,2	-18,1	4,75	-0,8	49	46,3
42.	-19,7	-18,1	-2,82	-0,8	51,9	46,3
43.	-20,1	-18,1	-4	-0,8	48,8	46,3
44.	-19,1	-18,1	0,63	-0,8	50,3	46,3
45.	-19,3	-18,1	0,39	-0,8	52	46,3
46.	-18	-18,1	1,49	-0,8	48,6	46,3
47.	-18,5	-18,1	1,37	-0,8	48,8	46,3
48.	-20,3	-18,1	-5,1	-0,8	47,7	46,3
49.	-15,7	-18,1	-1,1	-0,8	46,7	46,3
50.	-15,1	-18,1	2,94	-0,8	47,4	46,3
51.	-18,5	-18,1	0,12	-0,8	49,1	46,3
52.	-18,5	-18,1	-4,43	-0,8	51,3	46,3
53.	-21,2	-18,1	-6,35	-0,8	40,6	46,3

No.	X	Th_x	Y	Th_y	Z	Th_z
54.	-16,2	-19,4	5,3	-1,12	46,8	47,4
55.	-17,9	-19,4	0,55	-1,12	50	47,4
56.	-28,1	-19,4	2,31	-1,12	52,3	47,4
57.	-16,6	-19,4	3,06	-1,12	43,4	47,4
58.	-13,3	-19,4	0	-1,12	47,2	47,4
59.	-17,3	-19,4	-7,53	-1,12	48,6	47,4
60.	-19,1	-19,4	5,81	-1,12	49,7	47,4
61.	-18,3	-19,4	1,22	-1,12	50,6	47,4
62.	-20,2	-19,4	-0,24	-1,12	51	47,4
63.	-21,6	-19,4	8,04	-1,12	42,4	47,4
64.	-10,7	-19,4	1,8	-1,12	44,5	47,4
65.	-19,3	-19,4	-10,3	-1,12	51,2	47,4
66.	-16,4	-19,4	1,88	-1,12	49,7	47,4
67.	-19,6	-19,4	-4,12	-1,12	52,3	47,4
68.	-19,9	-16,3	-0,08	2,04	49,8	49,1
69.	-20,4	-16,3	-5,53	2,04	51,5	49,1
70.	-15,7	-16,3	5,96	2,04	44,6	49,1
71.	-15,7	-16,3	-0,94	2,04	46,3	49,1
72.	-11,2	-16,3	-0,94	2,04	48,1	49,1
73.	-18,9	-16,3	7,61	2,04	50,7	49,1
74.	-18,1	-16,3	1,77	2,04	49	49,1
75.	-21	-16,3	0,47	2,04	49,1	49,1
76.	-15	-16,3	9,61	2,04	45,9	49,1
77.	-12,7	-16,3	4,59	2,04	44,7	49,1
78.	-14,4	-16,3	-2,16	2,04	51,7	49,1
79.	-21,5	-16,3	4,51	2,04	53,5	49,1
80.	-20,1	-16,3	1,69	2,04	50,4	49,1
81.	-18,2	-16,3	2,2	2,04	50	49,1

No.	X	Th_x	Y	Th_y	Z	Th_z
82.	-17,8	-17,4	2,28	1,22	50,2	48,6
83.	-14,5	-17,4	5,77	1,22	44,3	48,6
84.	-13,9	-17,4	3,84	1,22	45	48,6
85.	-16,6	-17,4	-6,94	1,22	50,7	48,6
86.	-19,9	-17,4	9,38	1,22	50,9	48,6

Tabel A. 2. Selisih Data Sensor dengan *threshold* setiap sudut pada Tabel A.1.

No.	$\|X - Thx\|$	$\|Y - Thy\|$	$\|Z - Thz\|$
1.	3,135	5,63	3,85
2.	0,825	6,14	2,51
3.	1,485	2,45	0,79
4.	0,045	0,57	2,32
5.	0,515	1	2,95
6.	1,025	1,11	2,36
7.	3,765	1,47	2,87
8.	0,315	5,59	3,22
9.	3,455	2,72	1,06
10.	2,745	0,57	2,44
11.	3,765	6,14	1,73
12.	0,435	1,23	3,85
13.	1,455	1,31	1,15
14.	5,1	1,055	3,92
15.	5,77	4,115	2,87
16.	0,2	4,235	0,47
17.	6,16	3,485	5,02
18.	1,26	4,115	2,35
19.	1,1	2,355	2,86

No.	$\|X - Thx\|$	$\|Y - Thy\|$	$\|Z - Thz\|$
20.	0,98	1,295	5,53
21.	1,3	5,605	5,53
22.	6,16	2,585	2,87
23.	3,57	8,315	0,08
24.	0,79	8,315	0,79
25.	0,12	0,195	5,02
26.	0,51	0,745	1,88
27.	7,295	3,53	2,23
28.	6,235	3,61	4,59
29.	7,295	2,04	0,04
30.	5,725	2,94	2,19
31.	0,825	6,59	2,35
32.	1,055	1,38	0,74
33.	1,805	1,34	0,63
34.	0,435	6,59	4,59
35.	1,095	1,53	2,15
36.	0,515	1,45	2,78
37.	0,195	1,14	2,98
38.	1,765	1,92	2,04
39.	1,605	0,9	0,31
40.	0,195	4,67	0,32
41.	0,92	5,55	2,67
42.	1,55	2,02	5,62
43.	1,94	3,2	2,56
44.	0,96	1,43	4,01
45.	1,16	1,19	5,73
46.	0,1	2,29	2,36
47.	0,37	2,17	2,52

No.	$\|X - Thx\|$	$\|Y - Thy\|$	$\|Z - Thz\|$
48.	2,14	4,3	1,42
49.	2,41	0,3	0,4
50.	3,08	3,74	1,11
51.	0,34	0,92	2,79
52.	0,37	3,63	5,03
53.	3,08	5,55	5,72
54.	3,22	6,42	0,6
55.	1,53	1,67	2,61
56.	8,67	3,43	4,93
57.	2,75	4,18	3,94
58.	6,04	1,12	0,13
59.	2,08	6,41	1,28
60.	0,28	6,93	2,34
61.	1,06	2,34	3,28
62.	0,78	0,88	3,63
63.	2,23	9,16	4,93
64.	8,67	2,92	2,84
65.	0,08	9,16	3,79
66.	3,02	3	2,38
67.	0,23	3	4,93
68.	3,53	2,12	0,73
69.	4,08	7,57	2,42
70.	0,67	3,92	4,46
71.	0,67	2,98	2,73
72.	5,14	2,98	1,01
73.	2,59	5,57	1,67
74.	1,76	0,27	0,02
75.	4,63	1,57	0,05

No.	$\|X - Thx\|$	$\|Y - Thy\|$	$\|Z - Thz\|$
76.	1,37	7,57	3,2
77.	3,65	2,55	4,34
78.	1,88	4,2	2,65
79.	5,14	2,47	4,46
80.	3,8	0,35	1,36
81.	1,92	0,16	0,95
82.	0,41	1,06	1,59
83.	2,85	4,55	4,34
84.	3,51	2,62	3,6
85.	0,73	8,16	2,1
86.	2,53	8,16	2,3

[Halaman ini sengaja dikosongkan]

BAB VI
PENUTUP

Pada bab ini akan dibahas mengenai kesimpulan yang dapat diambil selama pengerjaan tugas akhir dari awal hingga selesai. Pada bab ini juga dapat menjawab pertanyaan yang dijabarkan pada Bab 1. Pembuatan tugas akhir ini pasti memiliki beberapa kelebihan dan kekurangan dari hasil yang telah dicapai dari pembuatan sistem. Semua kelebihan dan kekurangan tugas akhir ini juga akan dijabarkan pada bab ini. Untuk memperbaiki semua kelebihan dan kekurangan dari sistem ini, akan dijelaskan pada subbab saran.

6.1. Kesimpulan

Selama pengerjaan tugas akhir ini, dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Sistem mampu mendeteksi langkah dan mengukur jarak yang ditempuh dengan tingkat akurasi yang tinggi. Rata-rata untuk deteksi langkah 92,22% dan Jarak 86,64% ketika perangkat mikrokontroler digunakan di knee pad di lutut.
2. Tingkat presisi kalori pada sistem ini terhadap aplikasi pembanding yang bereputasi baik adalah sebesar 86,58%.
3. Dengan kondisi berjalan yang tidak menentu, proses penyimpanan data setiap 2 detik sekali di EEPROM dapat tersimpan 100% dibanding penyimpanan menggunakan modul eksternal menyebabkan modul rusak dan transmisi data ke aplikasi android yang mengakibatkkan *loss data*.
4. Perancangan jaringan nirkabel sederhana dengan memanfaatkan mikrokontroler arduino, modul *bluetooth*, sensor akselerometer dan handphone menggunakan aplikasi android.

6.2. Saran

Selama proses pengerjaan tugas akhir yang meliputi perancangan, implementasi dan uji coba, ditemukan beberapa kekurangan pada sistem. Beberapa saran yang dapat dilakukan untuk meminimalisasi kekurangan tersebut adalah sebagai berikut:

1. Proses pengolahan data dapat diolah secara cepat.
2. Menggunakan sensor *altitude*, *heart rate* dan GPS pada perangkat pemantau kalori.
3. Menggunakan sensor yang performanya lebih baik dan lebih akurat sehingga dapat mengurangi *noise* data yang dihasilkan dari sensor.
4. Menggunakan media penyimpanan yang besar sehingga data dapat disimpan lebih banyak.

DAFTAR PUSTAKA

[1] Analog, "ADXL 345," [Online]. Available: http://www.analog.com/en/products/mems/mems-accelerometers/adxl345.html#product-overview. [Diakses 17 Desember 2015].

[2] Arduino, "Arduino Pro Mini," [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardProMini. [Diakses 17 Desember 2015].

[3] Arduino, "EEPROM Write," [Online]. Available: https://www.arduino.cc/en/Reference/EEPROMWrite. [Diakses 19 Juni 2016].

[4] Wiki, "Bluetooth HC-06," [Online]. Available: https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To. [Diakses 13 Juni 2016]

[5] Zhao, N. 2010. Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer.

[6] Wikipedia, "Android," [Online]. Available: https://id.wikipedia.org/wiki/Android_(sistem_operasi). [Diakses 18 Desember 2015].

[7] Math. "Distance, rate and time," [Online]. Available: http://www.math.com/school/subject1/lessons/S1U2L3DP.html. [Diakses 28 Juni 2016].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Penulis, Achmad Fauzi Azmi dilahirkan di Malang 22 tahun silam. Penulis menempuh pendidikan mulai dari SDIT Fajar Hidayah (2000-2006), SMPN 9 Jakarta (2006-2009), SMAN 6 Jakarta (2009-2012). Pada tahun 2012, penulis diterima di strata satu Jurusan Teknik Informatika Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya angkatan 2012 yang terdaftar dengan NRP. 5112100192.

Pada beberapa acara kampus, penulis juga aktif menjadi sukarelawan TC Mengabdi pada tahun 2014-2015. Selain sebagai sukarelawan, penulis pernah mengikuti beberapa perlombaan, diantaranya PKM KC tahun 2012, PKM GT tahun 2013.

Kritik dan saran sangat diharapkan guna peningkatan kualitas dan penulisan selanjutnya. Untuk itu, silahkan kirim kritik dan saran ke : **fauziazmi@live.com**